JACKAL: A Protein Structure Modeling Package

Jason Z. Xiang

Columbia University &

Howard Hughes Medical Institute

Email: zx11@columbia.edu

July, 2002

Content

1. Abstract
2. Copyright
3. Introduction
4. Installation
5. Tutorial

1. Abstract

JACKAL is a package for protein structure modeling with BINGO as its graphic environment. JACKAL has the following capabilities:

1) comparative modeling based on single, composite or multiple templates;
2) side-chain prediction;
3) modeling residue mutation, insertion or deletion;
4) loop prediction;
5) structure refinement;
6) reconstruction of protein missing atoms;
7) reconstruction of protein missing residues;
8) prediction of hydrogen atoms
9) fast calculation of solvent accessible surface area;
10) structure superimposition;
11) manipulation of protein structure in torsional angle space.

JACKAL consists of about 90,000 lines of C++ code with the nest program as its core and is module enough to allow easy integration of new capabilities in the future. Most of the work has been done during my stay in the last one and half years at the Biochemistry Department of Columbia University and HHMI.

Availability:
JACKAL software and manual: http://trantor.bioc.columbia.edu/~xiang/jackal
JACKAL graphic environment: http://trantor.bioc.columbia.edu/~xiang/bingo

2. Copyright Notice

The program is distributed free of charge as it is to any users. The author holds no liabilities to the performance of the program.

Re-distribution of the program is permitted with the author's consent.

3 Introduction

The human genome project has produced about 700,000 protein sequences, among which only less than 20,000 have their structures determined either by x-ray or NMR. Protein structure modeling has become increasingly important in filling the huge gap between the number of available protein sequences and the number of available structures. Previous CASP experiments have demonstrated that knowledge-based protein structure modeling can often generate predictions with an accuracy equivalent to low-resolution x-ray structures. The objective of the structure-modeling package, JACKAL, is to provide user-friendly and efficient tools for building, refining, manipulating and reconstructing protein structures. The development of JACKAL package was undertaken to streamline the tools so that they can interoperate to perform complex computations.

The current version of JACKAL includes 8 structure prediction programs and many common utility tools:
nest for model building based on a given sequence-template alignment;
loopy for protein loop prediction;
scap for protein side-chain prediction;
profix for prediction of protein missing atoms and residues;
ctrip for prediction of protein backbone from CA atoms;
conref for protein structure refinement;
minst for protein structure minimization using smoothing energy technique;
addh for reconstruction of hydrogen atoms;
surface for calculation of solvent accessible surface;
nalgn for superimposition of two protein structures;
rotate for manipulation of protein structure in torsional angle space.

Some programs may have similar functions but differ in their algorithm and execution. For example, both the *scap* and *loopy* program can perform residue mutation. However, residue mutation in *loopy* comes with backbone conformational change while the backbone is fixed in *scap*; both *conref* and *minst* can minimize a protein structure, but they differ in that *conref* involves conformational sampling while *minst* uses steepest descent with a single conformation; both the *profix* and *addh* can reconstruct protons but differ in their approaches: *profix* uses a rotamer library while *addh* uses distance geometry to predict hydrogen atoms. The user decides which approach mostly suits his or her needs. When dealing with the same problem, different approaches pose their own pros and cons. For example, residue mutation with loopy may sound more reasonable but risks disturbing the backbone conformation.

There are two layers of classes in JACKAL. The first level contains the basic utility classes for, e.g., reading a PDB file, loading parameters, calculating solvent accessible surface, computing energies and performing standard Needleman-Wunsch sequence alignment. The basic utility classes have been compiled into a standalone utility application; at the second level are classes for prediction such as side-chain (*scap*) and loop prediction (*loopy*). The prediction classes on the second layer are built based on the

first layer. Different prediction classes interact with each other by library reference to ensure execution efficiency. The object-oriented characteristics of the C++ language make it easy for class reference and functionality expansion. The current version of the package has been ported and tested on SGI, Solaris and Linux platforms. Rather than incorporating all the classes into one monolithic program, JACKAL provides a set of relatively small programs that can be integrated into more complex applications.

The second layer of classes was mainly built to serve the JACKAL core program *nest*. The main task of *nest* is to build protein structures based on single, composite or multiple templates. It also has various options for alignment tuning and model refinement. Nest builds protein models based on the given alignment between query sequence and template. The model building is based on the artificial evolution method. Given an alignment between query sequence and template, the alignment can be considered as a list of operations such as residue mutation, insertion and deletion performed on the template structure. *Profix* and *ctrip* are used to check and fix any missing atoms in the template. The model building starts from the operation where the least energy cost is involved. Residue mutation is performed with *scap*, and residue insertion and deletion is handled by *loopy*. Each operation is finished with an energy minimization to remove atom clashes with *minst*. The final structure is then subjected to a more thorough energy refinement with *conref*.

The basic library design of loading engine used by JACKAL allows usage of several different parameter sets. At present we are supporting two sets of force field parameters (CHARMM22 and AMBER94 force fields) and three choices of rotamer libraries (small, medium and large rotamer libraries for both side-chain and backbone). Switching between different parameter set is easy and can be done by choosing different options at the command line.

4) Installation

4.1) Platform

JACKAL supports the following platform:

SGI, Sun Solaris and Intel Linux.

4.2) Download Software

4.2.a) Compressed tar-file for SGI v6.5 is available at:

http://trantor.bioc.columbia.edu/~xiang/jackal/jackal_sgi.tar.gz

4.2.b) Compressed tar-file for Sun Solaris is available at:

http://trantor.bioc.columbia.edu/~xiang/jackal/jackal_sun.tar.gz

4.2.c) Compressed tar-file for Intel Linux is available at:

http://trantor.bioc.columbia.edu/~xiang/jackal/jackal_linux.tar.gz

4.3) Install Software

Installation of the software on SGI, SUN and Linux platforms works in the same way. Take SGI as an example:

Step 1. Download jackal_sgi.tar.gz
Step 2. Unpack the compressed tar file:
gunzip jackal_sgi.tar.gz
tar xvf jackal_sgi.tar

This will create a new directory called jackal which contains the following files and directories:
doc/
addh/
conref/
ctrip/
loopy/
minst/
nest/
profix/
scap/
util/
library/
bin/
README

About the subdirectories:

doc is a directory containing relevant document files; the library directory contains all libraries used in JACKAL program; bin contains JACKAL executable programs; other directories contain tutorials and linked executable programs. For example, nest directory contains an executable program "nest", which is only a link to ../bin/nest, and example files how to use nest. The document of nest can be found in jackal/doc/Nestmanual.pdf or jackal/doc/Nestmanual.htm.

About jackal.dir

jackal.dir is a file which directs programs where to find libraries and pdb files. Further it checks license. It can be found in the jackal/bin directory. jackal.dir contains three settings: "pdb", "library" and "key". The setting of "pdb" is a list of directories following the "pdb:". The list of directories after "pdb:" tells program to search for pdb files in those directories sequentially. When you specify a pdb code, for example, you can either specify it as 1cbn.pdb or simply as 1cbn. JACKAL will first try to open 1cbn.pdb and then 1cbn file which ever exists.

The setting of "library" is a directory where JACKAL can find library files. The default setting under subdirectory bin is:

library: ../library

The library is located in jackal/library. When you run the JACKAL programs in a directory other than jackal/bin directory, you should set the jackal.dir file correctly so that library points to the correct library path.

The key is a password to activate JACKAL program. After you have downloaded the JACKAL package, you should send email to the author asking for the key.

How can the program nest find jackal.dir ? When the JACKAL program starts running, it first tries to read the file "jackal.dir" from the current directory. If it cannot find jackal.dir from the current working directory, the JACKAL program tries to find "jackal.dir" from the environment path with the name "JACKALDIR".

Step 3. Set the environment path

You can add the following two sentences to .cshrc file:
setenv JACKALDIR /your_home_directory/jackal/bin/jackal.dir
setenv PATH "$PATH": "/your_home_directory/jackal/bin"

For example, if your home directory is /home/xiang and you install jackal under your home directory:
setenv JACKALDIR /home/xiang/jackal/bin/jackal.dir
setenv PATH "$PATH":"/home/xiang/jackal/bin"

with jackal.dir in jackal/bin it looks like:
pdb:
    /hosts/razor/0/databases/pdb/

/pdb
library: /home/xiang/jackal/library
key: you-need-the-key-to-run

The .cshrc file can be found at your home directory. Open the file jackal.dir in jackal/bin and edit the pdb and library settings so that the pdb and library point to their right paths.

After you have added the above two lines to .cshrc, you need to type the command "source .cshrc" to activate the two sentences. Now you can run the jackal program from any directory.

5. Tutorial

5.1) nest
5.2) scap
5.3) loopy
5.4) profix
5.5) ctrip
5.6) addh
5.7) conref
5.8) minst
5.9) Util

5.1) nest

Introduction
Input Formats
Commands
Tutorial
Log File

Introduction

a) What is Nest?

Nest is a program for modeling protein structure based on a given sequence-template alignment. The program has the following capabilities: homology model building, composite model building, model building based on multiple templates, structure refinement, sequence-alignment tuning. The core of nest consists of nearly 80,000 lines of codes written in pure standard C++.

Some of the algorithms and methods used by Nest have been described in a number of papers from the Biochemistry Department of Columbia University in the last years and some will be covered in the coming papers, which are under preparation. The essence of the algorithms in the program will also be described briefly in this manual as well.

b) Capabilities of Nest

Nest is a program for building structures based on a given alignment between query sequence and template structure. Nest has the following capabilities:

1. Model building with artificial evolution
2. Sequence alignment tuning
3. Composite structure building
4. Model building based on multiple templates
5. Structure refinement

Model building with artificial evolution: Nest builds protein models based on a given alignment between query sequence and template. The model building is based on the artificial evolution method. Given an alignment between query sequence and template, the alignment can be considered as a list of operations such as residue mutation, insertion or deletion. Building a structure for the query sequence based on the template is thus a process of performing those operations. Each operation will disturb the template structure and thus involve an energy cost, either positive or negative. The model building starts from the operation with the least energy cost and so on. Each operation is finished with a slight energy minimization to remove atom clashes. The final structure is then subjected to more thorough energy minimization. The minimization is done in torsion angle space. The energy function consists of the following terms: van der Waals energy, hydrophobic,

electrostatics, torsion angle energy, hydrogen-bond network energy of the template, and statistical energy of a residue's solvent accessibility.

Sequence alignment tuning: Nest has different options to tune the sequence alignment such as gaps in helix or sheet regions, zig-zag gaps, or terminal gaps.

Composite structure building: Composite structure building is used in the case where the query sequence has several regions, each of which corresponding to different templates. Thus, the final model can be built based on their corresponding templates and fusing them into the final composite model.

Model building based on multiple templates: Multiple templates may be used in the case that the query sequence has more than one homologous template. Nest can build a model based on each of the homologous templates, thus multiple models can be built. The multiple models are then superimposed and the regions of high variability are then identified. For the region of high variability, nest tries all different conformation possibilities from other templates and seeks to identify the conformation with the lowest energy.

Structure refinement: The structure refinement module in nest can refine the built models in four levels: energy minimization of clashing atoms, refinement of insertion and deletion regions, refinement in all loop regions and refinement in all α/β regions. Energy minimization together with the removing of atom clashes are performed using the fast torsion angle minimizer together with smoothing energy techniques. Refinement in loop regions is done using the loopy program and refinement in side-chain conformation is performed by the side-chain program scap. Refinement of helix or sheet regions is done by a procedure similar to loopy while the hydrogen constraints in the regular secondary regions are applied so that the refinement will not disrupt the original hydrogen network. A hydrogen bond has the same definition as in the DSSP program.

Structure refinement is used to refine the original structure to be closer to the native, if possible. The refinement is done through energy minimization in helix, sheet and coil regions. There is also a standalone refinement program called conref and minst coming with the package.

Input Formats

1) Input format for single alignment block:
Nest adopts the pir alignment format. Each file can contain multiple alignment blocks. An alignment block is a region marked by the "#start" and "#end" symbols. A file can contain multiple alignment blocks. The simplest alignment block is composed of two pir alignments, corresponding to the query sequence and the template sequence respectively. The following is a simple example of a valid alignment file including one alignment block:

```
#start

!ex1.pir: a simple example about input format.

>P1;1hbaA
structureN:1hbaA:0:A:141:A
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFD--L----SH-GS
AQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAA
HLPAEFTPAVHASLDKFLASVSTVLT-SK-YR*

>P1;SEQ
sequence:myg:1: :150: :
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASE
DLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKIPVKYLEFISEAIIQVLQSK
HPGDFGADAQGAMSKALELFRNDMAAKYKLG*
#end
```

In the alignment file, lines beginning with "!" will be treated as comment line and thus ignored. A valid alignment block starts with "#start" and ends with "#end" symbols. If a file contains more than one alignment block, the "#start" and "#end" symbols must exist to separate different alignment blocks; otherwise, the "#start" and "#end" symbols can be omitted.

In any alignment block, there must be at least two lines starting with ">P1;". The line starting with ">P1;" is called the pir starting line, which is used to indicate a new pir alignment. The line just below the pir starting line is called the pir token line, which is used to indicate template or query information about the pir alignment.

In any alignment block, there must be one and only one pir alignment for structure and sequence respectively. The "structure" and "sequence" pir alignments are marked by the symbol "structure" and "sequence" beginning at the token line. For example, in the above alignment block, the first pir alignment is called structure pir and the second pir is called sequence pir alignment.

2) Structure and sequence token line

Besides the structure and sequence token lines, there is another token line called "composite token line", which will be described later in the composite and multiple template section. Structure and sequence token lines have the following format:

token:name:res start num:chain id:res end num: chain id

Where the token can be "structureX", "structure", "sequence-X", or "sequence". The one character "X" can be any character from a-z, A-Z or 0-9. The token of "structureX" is simply treated as "structure". It is kept to make Nest input format consistent with Modeller. The token "structure" indicates that the pir alignment is for template structure while the token "sequence" indicates that the pir alignment is for query sequence; the symbol "sequence-X" is used to indicate that the pir alignment is for query sequence and that the model built for the query sequence is represented by the character "X". This is useful when building composite models as will be explained in detail in the next section.

In the token line, a name must be given for the structure pir alignment; it can be omitted for the sequence pir alignment. However, if the name is missing for the sequence pir alignment, the model built for the query sequence will not be outputted to hard disk. At least one sequence/composite pir alignment must have its name given, i.e., that means at least one structure should be outputted. If the token is "structure", the name in the pir alignment is the pdb file of the template. As shown in the above example, "1hbaA" is the name of the template, and a file called "1hbaA.pdb" must exist in one of the pdb directories listed in the file jackal.dir. The jackal.dir file can be found under bin directory or pointed to by the environment variable JACKALDIR. The name in the sequence pir alignment is the name for the modeled structure of the query sequence that will be outputted to the hard disk. As in the above example, the sequence pir alignment has a name "myg", and the modeled structure for the query sequence will be outputted to the current directory as a file "myg_final.pdb". Other files of intermediates may also be outputted to the current directory with the name "myg_*.pdb", such as "myg_ini_model.pdb", which means the initial model before removing atom clashes.

The name in token line has a default appendix ".pdb". If the name does not end with ".pdb", then the name is appended with ".pdb".

The chain id indicates which chain that should be used in pir alignment. If the chain id is not specified, the default is the first chain. However, if it is specified, the chain id must be specified consistently, i.e., chain id after the "res start num" and "res end num" must be the same character.

Residue start and end numbers do not need to be specified because they can be found out by doing Needle-Wunsch alignment between the residues in structure pir alignment and the residues in the corresponding chain of the pdb file. However, if the start and end numbers are set, the nest program will then check if the start and end numbers are set correctly by comparing residues of the pir alignment and residues from the start to the end in the pdb file. If any residue name does not match, the start and end numbers will be ignored and the Needle-Wunsch alignment is performed to find out the correct start and

end residue numbers. The output of the Needle-Wunsch alignment will be outputted to the log file.

The start and end numbers in the sequence pir alignment do not need to be specified. If they are specified, only the start number is used to indicate the starting residue id of the model.

All the characters below the token line and the next starting pir line are considered as residues. Only standard amino acid residue name plus dash "-" are accepted; other characters are treated as dashes. Small capital characters are converted into large capital. The number of residues plus dashes in each pir alignment must be equal.

3) Identify errors in structure pir alignment

The residues in the structure pir alignment and in the pdb file may not match with each other. For example, some pdb structures may have missing residues while those missing residues exist in the pir alignment; or some residues exist in the pdb structure but are missing in the structure pir alignment. These unusual situations can be detected by aligning the residues in the structure pir alignment against the residues in the pdb files. If the residue exists in the structure pir alignment but does not exist in the chain of pdb file, it will be replaced by a dash "-". However, if the residue exists in the chain of pdb file but does not exist in the structure pir alignment, the residue will be inserted into the structure pir alignment, which will create a gap in the sequence pir alignment. If the residue in the chain of the pdb file does not match the residue in the structure pir alignment, the residue name in the structure pir alignment will then be changed to match that in the pdb file, though this situation rarely happens compared with the first two.

Below is an example of an alignment block:

```
>P1;an example to correct errors in structure pir alignment
structure:1cbn:::::::
TTCCPSSIVARSNFNVCRL--PGTPEALCATYT-CIIIPYATCPGDYAN
>P1;query sequence
sequence:test:::::::
RTCCFSSIVARSKFNVCRLPGTPEAL--CATYTGCIIIPGATCPGDYAN
```

The structure is 1cbn.pdb, which has 46 residues. Due to some unknown reasons, the residues in the structure pir alignment are not exactly equivalent to those in 1cbn.pdb file. When running nest, nest first makes Needle-Wunsch alignment between the sequence in the structure pir alignment and that in 1cbn.pdb file. Nest detects some errors and will print out the error message as follows:

```
**********warning!***********
the number of residues in the structre:1cbn is:45
the number of residues in the alignment:46
the number does not match...
performing Needleman-Wunsch alignment to find the corresponding residue
in pdb file and in pir file...
```

```
TTCCP-SIVA RSNFNVCRLP GTPEALCATY TGCIIIPGAT CPGDYAN
TTCCPSSIVA RSNFNVCRLP GTPEALCATY T-CIIIPYAT CPGDYAN
***** **** ********** ********** * *****_** *******
```

Warning! residue: S at position: 6 does not exist in PDB
deleting this residue from the alignment...

Warning! residue: G at position: 31 does not exist in alignment
trying to insert this residue to the alignment...

Warning! the residue names in pdb and alignment are different:G36
changing the residue name in alignment to that in pdb...

In the above Needle-Wunsch alignment, the first sequence comes from pdb file, and the second sequence comes from structure pir alignment. When retrieving residues from pdb file, only residues with coordinates are taken, i.e., the sequence from pdb file comes from coordinates, not from the sequence card.

By performing alignment between the sequences from the pdb file and from the structure pir alignment, the nest program identifies three errors: 1) residue "S" at position 6 in the structure pir alignment does not exist in 1cbn.pdb, thus it is replaced by a dash "-"; 2) residue 30, "G", in 1cbn.pdb does not exist in the structure alignment pir, thus the residue "G" is inserted into the structure pir alignment at the same time nest inserts a dash "-" at the corresponding position in the sequence pir alignment; 3) at position 36, residue "G" in 1cbn.pdb file is corresponding to residue "Y" in the structure pir alignment, thus the residue name "Y" is changed to "G".

So the real alignment used in nest will be:

```
>P1;1cbn
structure:1cbn::::
TTCCP-SIVARSNFNVCRL--PGTPEALCATYTGCIIIPGATCPGDYAN
>P1;cop
sequence:cop: :: :
RTCCFSSIVARSKFNVCRLPGTPEAL--CATYTGCIIIPGATCPGDYAN
```

The new alignment that is actually used in nest program will be recorded in the log file.

4) Default sequence alignment

If the number of standard residues plus dashes in a structure pir alignment is zero, the actual sequence used in the pir alignment will be all residues in the corresponding chain of the pdb file; if the number of residues plus dashes in a sequence pir alignment is zero, the actual sequence used in the sequence pir alignment will be identical to the structure pir alignment. If the number of residues in a structure pir alignment is zero while the

number of residues in the sequence pir alignment is not zero, then nest will do Needle-Wunsch alignment to align the sequences in the structure and sequence pir alignments.

For example,

```
>P1;an example to show default residue
structure:1hbaA:@:A:@:A
*
>P1;
sequence:hba:@::@::
*
```

The alignment block has zero number of residues for both structure and sequence alignment pirs. The structure pir alignment is equivalent to all residues in the chain A, and sequence pir alignment will also have the same residues as in the structure pir alignment. Thus the above alignment block is equivalent to the following:

```
>P1;1hbaA
structureN:1hbaA:@:A:@:A:
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFT
PAVHASLDKFLASVSTVLTSKYR

>P1;SEQ
sequence:myg:@::@::
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFT
PAVHASLDKFLASVSTVLTSKYR
```

Another example where the number of residues in a structure pir alignment are zero while in the sequence pir alignment are not zero:

```
>P1;1hbaA
structureN:1hbaA:0    :A:141  :A:::-1.00:-1.00
*
>P1;SEQ
sequence:myg:1    : :150  : :::-1.00:-1.00
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASE
DLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKIPVKYLEFISEAIIQVLQSK
HPGDFGADAQGAMSKALELFRNDMAAKYKLGFQG*
```

Since the number of residues in the structure pir alignment is zero, it will take all residues from the chain A of pdb file: 1hbaA.pdb. The residues in 1hbaA.pdb are:

```
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHG
KKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFT
PAVHASLDKFLASVSTVLTSKYR
```

After nest retrieves residues from 1hbaA.pdb, it will do Needle-Wunsch alignment between sequences in the structure and sequence pir alignments. The Needle-Wunsch

alignment uses blosum65 matrix with gap opening cost of 18 and gap extension cost 0.2. Thus, the above alignment block will be equivalent to:

```
>P1;1hbaA
structureN:1hbaA:0    :A:141  :A:::-1.00:-1.00
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF------DLSHGSA
QVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAH
LPAEFTPAVHASLDKFLASVSTVLTSKYR------
>P1;SEQ
sequence:myg:1    : :150  : :::-1.00:-1.00
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASE
DLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKIPVKYLEFISEAIIQVLQSK
HPGDFGADAQGAMSKALELFRNDMAAKYKELGFQG
```

5) Multiple alignment blocks

In one alignment file, multiple alignment blocks can be included. Each of the alignments is separated by "#start" and "#end" symbols. Below is an example consisting of two alignment blocks:

```
#start
>P1;first alignment block
structure:test1:*:A:*:A
eryenlfaqlndrreGAFVPFVTLGDPGIEQSLKIIDTL
iDAGADALELGVPFSDPLADG
>P1;
sequence:query1::::::::
----------mfkdGSLIPYLTAGDPDKQSTLNFLLAL
-DEYAGAIELGIPFSDPIADG
#end

#start
>P1;second alignment block
structure:test2:*:A:*:A
MERYENLFAQLNDRR--EGAFVPFVTLGDPGIEQSLKII
>P1;
sequence:query2::::::::
--------------MFKDGSLIPYLTAGDPDKQSTLNFL
#end
```

This alignment file contains two alignment blocks. Nest treats this alignment file equivalent to two independent alignment files, each containing one alignment block. In the first alignment block as shown above, some residues are in small capital, which will be changed to large capital when nest reads them in.

6) Composite template input format

In protein structure modeling, sometimes we may come across that the best model can be built by fusing two regions from different templates. For example, for a query sequence

A, we find a template B that is homologous to A. However, there may be a region in sequence A (let's call it region x) that has much higher sequence identity to a region in template C than to a region in template B, though template B is generally better for A than template C. In this case, the model for query sequence A can be best built by using template B while using template C for region x.

Below is an example of an alignment file for composite template building:

```
#start

>P1; template1
structure:tmp1:*:A:*:A:
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADG

>P1;T0122
sequence-a:xa::::::::
MFKDG-----SLIP-YLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADG

>P1;composite pir alignment
composite:xco:3:20:
aaaabbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1; template2
structure:tmp2:*:A:*:A
eRYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLAD

>P1;
sequence-b:xb::::::::
-----MFK------DGSLIPYLTAGDPDKQSTLNFLLALDEYAGAIELGIPFSDPIADG
#end
```

The above alignment file has two alignment blocks. Each alignment block has one structure and one sequence pir alignment. However, in the first alignment block, there is the third pir alignment marked by keyword "composite", which is called composite pir alignment. The composite pir alignment does not contain a residue sequence. It only contains the sequence of code names of the query models.

In composite template case, the code of each sequence pir alignment must be specified with one character with keyword such as "sequence-x". In the above alignment block, the first sequence pir alignment has been given a code of "a" and the second given a code of "b". Thus the models built for the first and second sequence pir alignments based on the templates tmp1.pdb and tmp2.pdb will have a code of "a" and "b", respectively. The sequence in the composite pir alignment defines all the regions except from position 5-12 are from template a. Region from position 5-12 will be replaced with the corresponding region in template b.

The positions 5-12 in the composite pir alignment cover residues from 4 to 6 in the query sequence, i.e., residue “GSL”, in the first alignment block. The three residue “GSL” can be found in the query sequence of the second alignment block, which is corresponding to positions from 16 to 18 in the sequence pir alignment in the second alignment block.

With this alignment file as input, nest will first build two models for the query sequence in the first and second alignment blocks, respectively. The two models will be outputted as file “xa_final.pdb” and “xb_final.pdb” respectively. The composite structure is then built by replacing the three residues “GSL” in xa_final.pdb with that from xb_final.pdb and outputted as “xco_final.pdb”.

The composite template can be used to replace any protein segment with another protein segment with the same residues, such as replacing coil with helix, helix with coil, etc. Composite templates can also be used to fuse two domains together. The domain-domain relative orientation can be pre-aligned or aligned by nest. For more detailed information, go to tutorial section.

In composite pir alignment, the name must be given; otherwise no composite structure will be built.

7) Multiple composite templates
Each alignment block can have multiple composite pir alignments. For example,

```
#start

>P1; template1
structure:tmp1:*:A:*:A:
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADG

>P1;T0122
sequence-a:xa::::::::
MFKDG-----SLIP-YLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADG

>P1;composite pir alignment
composite:xco1:3:20:
aaaabbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end

>P1;composite pir alignment
composite:xco2:3:20:
aaaaaaaaaaaaaaaaaaaaaaaaabbbbbbaaaaaaaaaaaaaaaaaaaaaa
#end

>P1;composite pir alignment
composite:xco3:3:20:20:20
aaabbbbbbbbaaaaaaaaaaaaaabbbbbbaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1; template2
```

```
structure:tmp2:*:A:*:A
eRYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLAD

>P1;
sequence-b:xb::::::::
-----MFK------DGSLIPYLTAGDPDKQSTLNFLLALDEYAGAIELGIPFSDPIADG
#end
```

In the above input file, the first alignment block contains three composite pir alignments. Each of the three composite pir alignments will be treated independently. In the second composite pir alignment, xco3, there are two regions which will be replaced by the segment from model b, which is built from the second alignment block.

8) Composite template token line

Composite token line has the following format:

composite:name:start:end:start:end:start:end

The token "composite" is used to indicate that the pir alignment is for composite. The "name" is used to output the composite final structure. The start and end integer values are used to tell the program how to decide their orientation when the segment is replaced. If there is one segment to be replaced, the composite token line should be like this:

composite:name:start:end:start:end

as shown in the xco1 and xco2 composite alignment above. However, if there are two regions to be replaced, the composite token line should contain two sets of start and end integer values:

composite:name:start:end:start:end:start:end

as shown in xco3 composite pir alignment above.

For each segment to be replaced there should be a corresponding set of start and end numbers to tell the program how to superimpose the two segments from different models.

For example, in the following example:

```
#start
>P1;1c29A
structure:1c29A:7:A:265:A:tryptophan synthase:salmonel::
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADGPTIQN
ANLRAFAAGVTPAQCFEMLALIREKHPTIPIGLLMYANLVFNNGIDAFYARCEQVGVDSV
LVADVPVEESAPFRQAALRHNIAPIFICPPNADDDLLRQVASYGRGYTYLLSRSGVTGAE
NRGPL--HHLIEKLKEYHAAPALQGFGISSPEQVSAAVRAGAAGAISGSAIVKIIEKNLA
SPKQMLAELRSFVSAMKAA*
>P1;T0122
sequence-a:xa::::::::
```

```
MFKDG------SLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADGKTIQE
SHYRALKNGFKLREAFWIVKEFRRHSST-PIVLMTYYNPIYRAGVRNFLAEAKASGVDGI
LVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVSLYGTTGAR
EEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSALVKIIGEK--
-GREATEFLKKKVEELLGI*
>P1;
composite:xc:10:15:
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa
#end

#start
>P1;1c29A
structure:1c29A:2:A:262:A:PDB::0.00:0.00
ERYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLAD
GPTIQNANLRAFAAGVTPAQCFEMLALIREKHPTIPIGLLMYANLVFNNGIDAFYARCE
QVGVDSVLVADVPVEESAPFRQAALRHNIAPIFICPPNADDDLLRQVASYGRGYTYLLS
RSGVTGAENRG--PLHHLIEKLKEYHAAPALQGFGISSPEQVSAAVRAGAAGAISGSAI
VKIIEKNLASPKQMLAELRSFVSAM*
>P1;T0122
sequence:xb:1:A:1:A:query::0.00:0.00
-----------MFKDGSLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIAD
GKTIQESHYRALKNGFKLREAFWIVKEFRRH-SSTPIVLMTYYNPIYRAGVRNFLAEAK
ASGVDGILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVS
LYGTTGAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSAL
VKIIGEKGREATEFLKKKVEELLGI*
#end
```

As shown in the above composite pir alignment, there is one region marked by b, which indicates that the region should be replaced with the corresponding segment from model b, which is the model built from the second alignment block. The region marked by code b in the composite pir alignment corresponds to the following 15 residues:

```
TAGDPDKQSTLNFLL
```

This 15-residue segment will be replaced by the corresponding segment from model b. Since model b and model a may have different orientations, usually we can not directly copy the 15-residue segment from segment b to segment a. The segment in model b should be first superimposed correctly before it is assembled on model a.

How to superimpose the two segments from model a and b ?
There are two choices: a) the user superimposes the two models manually and asks nest to assemble and fuse them; b) nest superimposes the two segments automatically;

a) The user superimposes the two segments. If the user chooses to superimpose the segments, the start and end values in the composite token line should be set zero. For example:

```
composite:xc:0:0:
```

In this case, the user can perform the following steps: first, build the two models xa_final and xc_final separately; second, superimpose the segments to be replaced in the two models using structure superimposition program; third, create a new input pir file with the two models as templates, for example.

```
#start
>P1;1c29A
structure:xa:7:A:265:A:tryptophan synthase:salmonel::
MFKDG------SLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADGKTIQE
SHYRALKNGFKLREAFWIVKEFRRHSST-PIVLMTYYNPIYRAGVRNFLAEAKASGVDGI
LVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVSLYGTTGAR
EEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSALVKIIGEK--
---GREATEFLKKKVEELLGI*
>P1;T0122
sequence-a:::::::::
MFKDG------SLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADGKTIQE
SHYRALKNGFKLREAFWIVKEFRRHSST-PIVLMTYYNPIYRAGVRNFLAEAKASGVDGI
LVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVSLYGTTGAR
EEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSALVKIIGEK--
---GREATEFLKKKVEELLGI*
>P1;
composite:xc:0:0:
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa
#end

#start
>P1;1c29A
structure:xb:2:A:262:A:PDB::0.00:0.00
-----------MFKDGSLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIAD
GKTIQESHYRALKNGFKLREAFWIVKEFRRH-SSTPIVLMTYYNPIYRAGVRNFLAEAK
ASGVDGILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVS
LYGTTGAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSAL
VKIIGEKGREATEFLKKKVEELLGI*
>P1;T0122
sequence::1:A:1:A:query::0.00:0.00
-----------MFKDGSLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIAD
GKTIQESHYRALKNGFKLREAFWIVKEFRRH-SSTPIVLMTYYNPIYRAGVRNFLAEAK
ASGVDGILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVS
LYGTTGAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSAL
VKIIGEKGREATEFLKKKVEELLGI*
#end
```

In the above input alignment file, the two templates are model structures with proper orientation. Please note that the sequences in the structure and sequence pir alignments are identical, and the start and end value in the composite token line are both zero.

With the two models properly superimposed, nest is going to fuse the ends of the 15-residue segment. It takes the following steps: first, build the model for the sequence pir alignment, since the sequence and structure pir alignments have the same sequence, the model for the sequence pir alignment is a copy of its template; second, transfer the 15-residue segment conformation from model b to model a; third, fuse the 15-residue segment ends within model a. The third step is essential because the transferred conformation from model b may violate bond angle and length constraints in model a. The fusing step is to connect residues of the two ends in the 15-residue segment: try to connect L (residue previous to the segment but not in the segment) with T (T is the head of the segment) and L (L is the tail of the segment) with A (residue just after the segment). If the fusion is not successful, then it moves to the inner body of the segment. For example, if L and T could not be connected (e.g., the two residues are two far away), then try to connect L with A; and so on. If non of this is successful, then the segment replacement is discarded and vice versa for the fusion at the segment tail.

b) nest superimposes the two segments. If the user lets nest superimpose the segments, the user must tell nest how to superimpose the segments. Each composite segment should have a start and end value specified.

b.1) superimpose beyond the segment itself. In the above case, the composite token line looks like:

```
composite:xc:10:15:
```

The start and end values are 10 and 15 respectively. The start and end values specify two regions based on which the two models can be superimposed. When the start value >0, the region is from the head of the segment (not including the head) to the N terminal of the model covering a number of residues specified by the absolute value of start. When the start value < 0, the region is from the head (including head) to the C terminal covering a number of residues specified by the absolute value of start; if the end value is >0, the second region is from the tail of the segment (not including tail) to the C terminal covering a number of residues specified by the absolute value of end. If the end value is <0, the second region is from the tail of the segment (including tail) to the N terminal covering a number of residues specified by the absolute value of end.

The two regions specified by the start and end value will generally direct how the two segments of different models can be possibly superimposed.

For example, the 15-residue segment is from residue 12-26 in model a. The first region specified by the start value of 10 means the region from residue 2 to 11, covering 10 residues; and the second region of the end value 15 means the region from residue 27 to 51 covering 15 residues. So the nest program will superimpose the two models based on the two aligned regions: from residue 2 to 11 and from residue 27 to 51.

b.2) superimpose within the segment itself. If the user would like to superimpose the two segments directly, the following composite token line could be used:

```
composite:xc:-15:0:
```

or

```
composite:xc:0:-15:
```

or

```
composite:xc:-7:-8:
```

All the above three composite token line examples have the same meaning, i.e., superimposing the segment itself.

The first line says the start value is –15 and the end value is 0. Since the start value is <0, it means the region from the head to the C terminal covering 15 residues, which is identical to the length of the segments; the end value is zero, which means there is no second region. So the whole superimposition is only the first region, which is equivalent to the whole segment. The second composite token line is similar to the first one.

The third line says the first region is from the head to the C terminal covering 7 residues and the second region is from the tail to the N terminal covering 8 residues, the two regions when combined together are equivalent to the whole 15-residue segment.

9) Multiple template input format

Multiple-template handling is used when several homologous templates are identified for the query sequence. By aligning these templates, the most variable region is usually located in the loop region and variation in the regular secondary regions is usually much smaller. Instead of building a model based on one template, the model for the query sequence can be built based on multiple templates. For the region of high variation, conformations from different templates are tried on the model and that with the lowest energy or highest sequence identity is then selected to replace the original conformation on the model. Thus the model built might be more accurate than based on only one template. Multiple-template handling adopts the formula of composite template handling, however with dash "-" to indicate the region where conformations from all other templates are tried.

Below is an example of an alignment file for multiple template handling:

```
#start
>P1; template1
structure:tmp1:*:A:*:A:
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADG
>P1;T0122
sequence-a:::::::::
MFKDG-----SLIP-YLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADG
>P1;composite pir alignment
composite:xco::::
```

```
aaaa--------aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1; template2
structure:tmp2:*:A:*:A
eRYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLAD
>P1;
sequence-b:::::::::
-----MFK------DGSLIPYLTAGDPDKQSTLNFLLALDEYAGAIELGIPFSDPIADG
#end

#start
>P1; template3
structure:tmp3:*:A:*:A
MERYENLFAQLNDRR-EGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPL
>P1;
sequence-c:::::::::
------MFK------DGSLIPYLTAGDPDKQSTLNFLLALDEYAGAIELGIPFSDPIAD
#end
```

In the above alignment file, there are three alignment blocks. Thus three models will be built for each of the alignment blocks. In the first alignment block, there is a composite pir alignment. The composite pir alignment has dash "-" covering region from position 5 to 12, which means the region equivalent to three residues "GSL" may be replaced by any conformation of the three models depending on energy/sequence identity.

Since there are no names given to the three sequence pir alignments, no output will be made for them. The composite pir alignment has been given a name called "xco" so that a file "xco_final.pdb" will be written down to hard disk.

Commands

For all the programs included in JACKAL package, you can find their usage by running the program with no arguments.

After unpacking jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/nest. This directory contains all examples used in this tutorial for running nest program. Go to this directory and type:

$nest

nest will print out message about its usage as following:

```
*******************************************************
nest is a homology model building program with the
following capabilities:
A. model building based on the given alignment
B. construction of composite structure
C. model building based on multiple templates
D. structure refinement
questions? please refer to Dr.Jason Z. Xiang at
jsxzx@yahoo.com.
*******************************************************
Usage: nest -seed num -fast num  -tune num -log str
            -opt num -nopt num  -out num file.pir
-fast     fast mode.from 0-5; default is 3.with 5 fastest
-tune     alignment tuning, from 0-3; defautl is 0.
          0,no alignment tuning
          1,remove gaps in secondary strucutre
          2,remove zig-zag gaps  plus -tune 1;
          3,remove terminal gaps plus -tune 22;
-opt      refine mode, from 0-4; default is 1.
          0, no minimization at all;
          1, remove atom clashes;
          2, refine in insertion and deletion regions,
             usually loop regions;
          3, refine in all loop regions;
          4, refine in all loop and secondary regions
-out      output mode, from 0-2, defautl is 1.
          0, only final structures written down;
          1, intermediate structures also outputted;
          2, more intermediate output
-seed     set seed number, defautl is 18120.
-nopt     number of rounds of refinement.default is 1.
-log      log file.default is standard screen output
file.pir  pir alignment file
```

When you run nest, you type: nest plus option plus the integer value for the option plus the alignment file. If there is something wrong with your command, nest will exit and print out an error message. The above options can be omitted, where nest will use the default values; or several options can be used at the same time.

fast option
The option "-fast num" is used to decide how fast the nest program should run. The fast mode has 6 levels from 0 to 5 with 5 as the fastest mode. In the case of mode 5, nest program will try to finish the model building as fast as possible, thus using less conformation sampling and energy minimization steps. The conformational sampling and energy minimization are required in residue mutation, deletion, insertion and searching for the least-energy-cost operations. With more CPU time spent, the final model usually has lower energy. When nest is running in default mode, level 3, the nest program will first mutate the residues that involve less than 10 kcal/mol energy cost; the second step is to insert or delete residues based on a list of operations, which is presorted according to the energy cost estimations. Each insertion and deletion will be performed with 30 random generated conformations. The final structure will have the remaining residues mutated and the atom-atom clashes removed.

tune option
The option "-tune num" is used to turn on or off the alignment tuning in the nest program. When the num >0, the tune is on; when the num is 0, the tune is off. The default value is tuning off.

When tune is 1, the nest program will remove insertion and deletions in the secondary structure region to its neighboring loop regions. The insertion/deletion is removed to the loop region whichever is closer to it.

When the tune option is 2, the alignment-tuning module will try to smooth out regions with zig-zag insertion and deletions in addition to removing gaps in regular secondary regions. For example, the following alignment has insertion and deletion zigzagged:

```
>P1;1hbaA
structureN:1hbaA:0:A:141:A:::
AEALERMFLSFPTTKTYFPHFD--L-S-H-GSAQVKGHGKKVADALTN

>P1;SEQ
sequence:test:1::150::::
QEVLIRLFKGHPETLEKFD-KFKHLK-E-EMKEDLKKHGNTVLTALGG
```

The alignment-tuning module in nest will re-adjust the above alignment to the following:

```
>P1;1hbaA
structure:1hbaA:A:0:A:141
AEALERMFLSFPTTKTYFPHFD--LSH---GSAQVKGHGKKVADALTN

>P1;SEQ
sequence:test::1::150
```

```
QEVLIRLFKGHPETLEKFDKFK-HLKE--EMKEDLKKHGNTVLTALGG
```

The reason to tune the above alignment is that the zigzag alignment will delete one residue from the template and then add a new residue back or vice versa. It is equivalent to residue mutation in most cases.

Besides zigzag alignment tuning, nest also smooth out gaps occurring in the C or N terminals using option 3. Option 3 also includes option 1 and 2 automatically. For example:

```
>P1;1hbaA
structureN:1hbaA:0    :A:141  :A:::-1.00:-1.00
AEAL--ERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTN
>P1;SEQ
sequence:myg:1    : :150  : :::-1.00:-1.00
QEVLAAIRLFKGHPETLEKFDKHLKEEMKEDLKKHGNTVLTALGG
```

After turning on alignment tuning with option 3, the new alignment will be:

```
>P1;1hbaA
structure:1hbaA:A:0:A:140
--AEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTN
>P1;myg
sequence:myg: :1: :150
QEVLAAIRLFKGHPETLEKFDKHLKEEMKEDLKKHGNTVLTALGG
```

The reason that nest offers this option is that insertion around terminals is energetically efficient to migrate to terminals instead of looping out.

You can turn off all tuning in nest by option "-turn 0".

Nest offers 4 levels of tuning from 0-3. If tune option is not from 0-3, it will print out an error message,

seed option
Seed option is just to give a random seed number to the nest program. The default is 18120. You can give any seed number using this option, as for example:

$nest –seed 102 ex1.pir

The above example sets the random seed number in nest to be 102. Different seed numbers sometimes influence the model with rmsd around 0.2 Å.

out option
Out option is used to direct the nest program to how the models for the query sequence should be outputted, such as, if the initial or intermediate models should be written down to hard disk.

When "-out 0" is used, the nest program will only write down the final models; when "-out 1" is used, the nest program will also write down intermediate structures; when "-out 2" is used, the nest program will output all intermediate structures in the process of refinement and composite model building. The intermediate and alternative models depend on the options used in running the nest program and the alignment files.

Suppose we have an example alignment file s.ali:

```
!example of output
>P1;
structure:s:*:A:*:A
eryenlfaqlndrreGAFVPFVTLGDPGIEQSLKIIDTL
iDAGADALELGVPFSDPLADG

>P1;T0122
sequence:x::::::::
----------mfkdGSLIPYLTAGDPDKQSTLNFLLAL
-DEYAGAIELGIPFSDPIADG
```

if you run the program with command:

$nest –out 0 s.ali

the only output will be x_final.pdb, which has atom clashes removed.

if you run nest with command:

$nest –out 1 s.ali

There will be three structure outputs: the final model and the intermediate structure. The final model is x_final.pdb, and the intermediate model will be x_ini_model.0.pdb and x_ini_min.1.pdb. The intermediate model is the structure after performing residue mutation, insertion and deletion according to the alignment table but before doing initial energy minimization to remove atom clashes. The x_ini_min.0.pdb is the model after atom clashes removed. In the above example, it is equivalent to the x_final.pdb.

However, if you run nest on s.ali with command:

$nest –out 2 –opt 2 s.ali.

The output from nest will be model x_final.pdb, intermediate model x_ini_model.0.pdb, x_ini_min.1.pdb and other intermediate structures with names similar to x_refine_loop.2.pdb. The refinement is performed on model x_final.pdb.

When the alignment file includes multiple alignment blocks, each alignment block will be treated independently with all the command options applied. When the alignment block also includes composite pir alignment, the command options will also be applied to the composite models where intermediate models come from composite structures.

If you do not like to have alternative or intermediate outputs for one particular alignment block, you can do so by setting the name of that sequence pir alignment empty.

opt option

The opt option is used to optimize and refine the model for the query sequence. There are 5 levels of refinement with 4 as the most thorough refinement and 0 as no refinement at all. The default level is set to 1. The option 1 is used to remove atom clashes. The option 2 is to refine regions where insertion and deletion occur, usually in loop regions. When refinement is applied to a specific region, nest will first sample a large number of conformations around the original conformation and discard those conformations with rmsd larger than 2.0A from the original conformation. So refinement is always performed with constraints around the original structure framework. However, the constraints can be relaxed by several rounds of refinement as explained below. In the process of refinement in helix and sheet regions, the hydrogen bonding network from the model is applied so that the final structure after refinement will also observe the original regular secondary structure regions. However, in the refinement of loop regions, the original loop regions may develop into helix or sheet depending on the energy criteria, though this does not usually happen.

Each conformation sampled will be subjected to our fast torsion space minimizer with smoothing energy technique.

nopt option

The nopt option is used to decide how many rounds of refinement are required. In the first round of refinement, the refinement feels pressure of framework constraints from the original structures; in the second round of refinement, the refinement is performed based on the output from the first round, so that the framework constraint pressure is relaxed compared with that in the original structure. Though in each round of refinement, the sampled conformation is restricted to within 2.0A from their starting conformation, this restriction will be more and more relaxed compared to the original structure with more rounds of refinement. However, it does not necessarily mean more rounds of refinement, usually further away from the original starting point, will bring the structure closer to the native, though energetically it should be lower.

log option

Log option is used to specify the log file output. If the log file is not specified, the output will be on the screen.

Tutorial

1) **Example 1: model building based on the alignment.**

Please go to the jackal/nest directory, there is a pir alignment file called ex1.ali. This file contains one alignment block with the template structure 1hbaA.pdb and the output name "myg" for the query sequence. The final structure will be myg_final.pdb.

The input alignment file ex1.ali:

```
>P1;1hbaA
structureN:1hbaA:     :A:   :A:::-1.00:-1.00
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFD--L----S
H-GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSH
CLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLT-SK-YR*
>P1;SEQ
sequence:ex1:     : :   : :::-1.00:-1.00
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKF-DKFKHLKSEDE
MKASEDLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKIPVKYLEFISE
AIIQVLQSKHPGDFGADAQGAMSKALELFRNDMAAKYKLG*
```

Under jackal/nest directory, type the command:

$nest ex1.ali

You should be able to build a model called myg_final.pdb and two other intermediate structures: myg_ini_model.0.pdb and myg_ini_min.1.pdb. The output log file will go to screen and inform you what is going on inside the nest program.

In ex1.ali, there is no start and end residue numbers specified. The nest program will try to find it out anyway.

You may also try different options such as:

$nest –out 2 –opt 4 –nopt 2 –tune 3 ex1.ali

The nest program will also write down a copy of the alignment file that is actually used by nest: ex1.ali~

2) **Example 2: multiple alignment blocks in one input file.**

You can also find another example input file: ex2.ali. This alignment input file contains 2 alignment blocks. Nest treats each of them independently. This is equivalent to run nest two times independently on the separate alignment blocks.

In ex2.ali, the template structure is 1c29 and the outputs for the two alignment blocks are ex2a and ex2b. Below is the ex2.ali input file:

```
!this example contains multiple alignmetn blocks.
!nest treats each of them independently.
!
#start
>P1;1c29A
structure:1c29:7:A:265:A:tryptophan synthase:salmonel::
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADG
PTIQNANLRAFAAGVTPAQCFEMLALIREKHPTIPIGLLMYANLVFNNGIDAFYA
RCEQVGVDSVLVADVPVEESAPFRQAALRHNIAPIFICPPNADDDLLRQVASYGR
GYTYLLSRSGVTGAENRGPL--HHLIEKLKEYHAAPALQGFGISSPEQVSAAVRA
GAAGAISGSAIVKIIEKNLASPKQMLAELRSFVSAMKAA*
>P1;T0122
sequence:ex2a::::::::
MFKDG------SLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADG
KTIQESHYRALKNGFKLREAFWIVKEFRRHSST-PIVLMTYYNPIYRAGVRNFLA
EAKASGVDGILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTT
GFVYLVSLYGTTGAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKE
GANGVVVGSALVKIIGEK---GREATEFLKKKVEELLGI*
#end

#start
>P1;1c29A
structure:1c29:2:A:262:A:PDB::0.00:0.00
ERYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFS
DPLADGPTIQNANLRAFAAGVTPAQCFEMLALIREKHPTIPIGLLMYANLVFNN
GIDAFYARCEQVGVDSVLVADVPVEESAPFRQAALRHNIAPIFICPPNADDDLL
RQVASYGRGYTYLLSRSGVTGAENRG--PLHHLIEKLKEYHAAPALQGFGISSP
EQVSAAVRAGAAGAISGSAIVKIIEKNLASPKQMLAELRSFVSAM*
>P1;T0122
sequence:ex2b:1:A:1:A:query::0.00:0.00
-----------MFKDGSLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFS
DPIADGKTIQESHYRALKNGFKLREAFWIVKEFRRH-SSTPIVLMTYYNPIYRA
GVRNFLAEAKASGVDGILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERL
KVIDDMTTGFVYLVSLYGTTGAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKR
EHVVSLLKEGANGVVVGSALVKIIGEKGREATEFLKKKVEELLGI*
#end
```

type the command:

$nest –log ex2.log ex2.ali

3) Example 3: Sequence alignment tuning

Ex3.ali is the same alignment file as ex1.ali:

```
>P1;1hbaA
structureN:1hbaA:    :A:  :A:::-1.00:-1.00
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFD--L
```

```
----SH-GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRV
DPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLT-SK-YR*
>P1;SEQ
sequence:ex3:    : :  : :::-1.00:-1.00
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKF-DKFKHL
KSEDEMKASEDLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKI
PVKYLEFISEAIIQVLQSKHPGDFGADAQGAMSKALELFRNDMAAKYKLG*
```

However, with the command:

$nest –tune 3 ex3.ali

The model might be different from that obtained in example 1 because it removes the potential zig-zag gaps. When nest is running, it will write an alignment file called ex3.ali~. This is the actual alignment used by the program. In comparing ex3.ali~ and ex3.ali one can find out the difference the tune option has made. The following is the ex3.ali~ file written down by nest program:

```
#start
>P1;1hbaA
structure:1hbaA:A:0:A:0
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFD
--L----SH-GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLH
AHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVST
VLTSKYR--
>P1;ex3
sequence:ex3: :0: :0
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKFDKFK
-HLKSEDEMKASEDLKKHGNTVLTALGGILKKKGHHEAELTPLAQSH
ATKHKIPVKYLEFISEAIIQVLQSKHPGDFGADAQGAMSKALELFRN
DMAAKYKLG
>P1;1hbaA; secondary structure
structure:1hbaA:A:0:A:0
---hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh-hhhhhh-----
------------hhhhhhhhhhhhhhhhhh----hhhh-hhhhhhh
hh------hhhhhhhhhhhhhhhhh------hhhhhhhhhhhhhhh
hh-------
#end
```

where the last pir is the secondary structure information of the template with ‘h’, ‘e’ and ‘-‘ for helix, sheet and coil, respectively.

You can also try different tuning options:

$nest –tune 1 ex3.ali

or

$nest –tune 2 ex3.ali

4) Example 4: Structure refinement with model building

The ex4.ali file is the same as ex1.ali except that the output file name is different. However, when you run the example using the command:

$nest –opt 2 –nopt 1 ex4.ali

The nest program will first build a model based on the alignment and then optimize the insertion and deletion regions. The final structure will be ex4_final.pdb.

or

$nest –opt 3 –nopt 1 ex4.ali

This command will optimize all insertion and deletion regions including all loop regions. The number of rounds of optimization is 1, which means that the final structure will be within 2.0A rmsd from the initial model.

$nest –opt 4 –nopt 1 ex4.ali

This command will optimize the initial model in all regions with 2.0A rmsd constraints.

However, in the case of multiple alignment blocks, if the sequence pir alignment does not have its name given, i.e., the model built for the sequence pir does not need to be outputted, the model will not be subjected to refinement beyond energy minimization removing atom clashes.

For example, in the following alignment input file:

```
#start
>P1;1b0u
structureX:1b0u:5:A:262:A: : : :
----NKLHVIDLHKRYG----GHEVLKGVSLQARAGDVISIIG-SSGSGKS-
TFLRCINF
>P1;T121
sequence:ex4b1:@:@:@:@: : : :
------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-
LRMIAG
#end

#start
>P1;1f3o
structureX:1f3o:1:A:232:A: : : :
--MIKLKNVTKTYKMGEEIIYALKNVNLNIKEGEFVSIMG-PSGSGKS-
TMLNIIGCLDK
>P1;T121
```

```
sequence::@:@:@:@: : : :
---VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-
LRMIAGLEE
#end

#start
>P1;1g6h
structureX:1g6h:0:A:257:A: : : :
---TMEILRTENIVKYFG----EFKALDGVSISVNKGDVTLIIG-PNGSGKS-
TLINVIT
>P1;T121
sequence:ex4b2:@:@:@:@: : : :
-------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-
LRMIA
#end
```

The sequence pir alignment in the second alignment block does not have the name given, so no output is required for the model, which was built for this sequence alignment. If you run the input file with this command:

$nest –opt 2 ex4b.ali

The nest program will first build 3 models for the three alignment blocks, then each of the three models will be subjected to energy minimization protocol removing atom clashes. However, since no output is required for the second alignment block, the refinement in the insertion and deletion regions will be only applied to the first and the third models. This is enforced because the model with no name associated with it is considered as temporary model. That model has no output requirement usually happens in the case of composite building case, where the only interested structure is the composite model.

5) Example 5: Structure refinement without model building

If you have a protein structure and want to do a refinement, where no model building is required, you can do this as shown in ex5.ali. The input file is the following:

```
>P1;1hbaA
structureN:1hbaA:    :A:  :A:::-1.00:-1.00
*
>P1;SEQ
sequence:ex4:    : :  : :::-1.00:-1.00
*
```

The input file has empty sequence for both structure and sequence pir alignments. So the model for the sequence pir alignment will be the copy of the template. You can refine the model with the same option as shown in example 4. However, since there is no insertion and deletion regions, the –opt 2 will be equivalent to –opt 1.

6) Example 6: Build composite structures from alternative alignments under the same templates.

The input file of example 6 is ex6.ali, which is as follows:

```
#start
>P1;1c29A
structure:1c29:2:A:262:A:PDB::0.00:0.00
ERYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLAD
GPTIQNANLRAFAAGVTPAQCFEMLALIREKHPTIPIGLLMYANLVFNNGIDAFYARCE
QVGVDSVLVADVPVEESAPFRQAALRHNIAPIFICPPNADDDLLRQVASYGRGYTYLLS
RSGVTGAENRG--PLHHLIEKLKEYHAAPALQGFGISSPEQVSAAVRAGAAGAISGSAI
VKIIEKNLASPKQMLAELRSFVSAM*
>P1;T0122
sequence-a::1:A:1:A:query::0.00:0.00
-----------MFKDGSLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIAD
GKTIQESHYRALKNGFKLREAFWIVKEFRRH-SSTPIVLMTYYNPIYRAGVRNFLAEAK
ASGVDGILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVS
LYGTTGAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSAL
VKIIGEKGREATEFLKKKVEELLGI*
>P1;
composite:ex6:0:20
bbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1;1c29A
structure:1c29:7:A:265:A:tryptophan synthase:salmonel::
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADGPTIQ
NANLRAFAAGVTPAQCFEMLALIREKHPTIPIGLLMYANLVFNNGIDAFYARCEQVGVD
SVLVADVPVEESAPFRQAALRHNIAPIFICPPNADDDLLRQVASYGRGYTYLLSRSGVT
GAENRGPL--HHLIEKLKEYHAAPALQGFGISSPEQVSAAVRAGAAGAISGSAIVKIIE
KNLASPKQMLAELRSFVSAMKAA*
>P1;T0122
sequence-b:::::::::
MFKDG------SLIPYLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADGKTIQ
ESHYRALKNGFKLREAFWIVKEFRRHSST-PIVLMTYYNPIYRAGVRNFLAEAKASGVD
GILVVDLPVFHAKEFTEIAREEGIKTVFLAAPNTPDERLKVIDDMTTGFVYLVSLYGTT
GAREEIPKTAYDLLRRAKRICRNKVAVGFGVSKREHVVSLLKEGANGVVVGSALVKIIG
EK---GREATEFLKKKVEELLGI*
#end
```

Example 6 is the case with two alignment blocks, where the templates are the same while the alignments are different. The structures of the two different alignments have large difference at their N-terminals. The program is to replace the first 14 residues in the N terminal in model a with that in model b. The sequence of these 14 residues is:

```
MFKDGSLIPYLTAG
```

Other parts of model a are kept unchanged. Since the names for the two sequence pir alignments are empty, so no outputs are required for model a and b. The final output is the composite structure called ex6_final.pdb.

Since the start and end value in the composite token is 0 and 20 respectively, the first region does not exist and the second region covers 20 residues from the tail of the segment to the C terminal in model a, i.e, the orientation of the 14 residues in model a and b can be determined by superimposing the 20 residues:

```
DPDKQSTLNFLLALDEYAGA
```

This 20-residue segment also exists in b. So the superimposition can be performed. If the 20-residue segment does not exist in model b, the composite structure building will fail.

Run the example with the command:

$nest ex6.ali

or if you also want to refine the composite structure in the insertion and deletion regions, run with this command:

$nest –opt 2 –nopt 2 ex6.ali

7) Example 7: Build multiple-region composite structures from different templates

The following is the input alignment file ex7.ali. There are three alignment blocks with the composite pir alignment in the first alignment block. The composite pir alignment has two regions in model a to be replaced with segment from models b and c.

```
#start
>P1;1b0u
structureX:1b0u:5:A:262:A: : : :
----NKLHVIDLHKRYG----GHEVLKGVSLQARAGDVISIIG-SSGSGKS-TFLRCINF
LEKPSEGAIIVNGQNINLVRDKDGQLKVADKNQLRLLRT-RLTMVFQHFNLWSHMTVLEN
VMEAPIQVLG-LSKHDARERALKYLAKVGIDERAQGKYPVHLSGGQQQRVSIARALAMEP
DVLLFDEPTSALDPELVGE-VLRIMQQLAEE-GKTMVVVTHEMGFARHVSSHVIFLHQGK
IEEEGDPEQVFGNPQSPRLQQFLKGSLKKLE
*
>P1;T121
sequence-a:ex7a:@:@:@:@: : : :
------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAG
LEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDN
IAFPLKL-RK-VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKP
QVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGV
LQQVGSPDEVYDKPAN--------TFVAGF-
*
```

```
>P1;
composite:ex7:20:20:20:20
bbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaccccccccccccccccccccccaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1;1f3o
structureX:1f3o:1:A:232:A: : : :
--MIKLKNVTKTYKMGEEIIYALKNVNLNIKEGEFVSIMG-PSGSGKS-TMLNIIGCLDK
PTEGEVYIDNIKTND----------LDDDELTKIRRDKIGFVFQQFNLIPLLTALENVEL
PLIF-KYRGAMSGEERRKRALECLKMAELEERFANHKPNQLSGGQQQRVAIARALANNPP
IILADEPTGALDSKTGEKIM-QLLKKLNEEDGKTVVVVTHDINVAR-FGERIIYLKDGEV
EREEK----------------LRGF-----
*
>P1;T121
sequence-b:ex7b:@:@:@:@: : : :
---VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAGLEE
PSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDNIAF
PLKL-RK---VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKPQ
VFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGVL
QQVGSPDEVYDKPAN--------TFVAGF-
*
#end

#start
>P1;1g6h
structureX:1g6h:0:A:257:A: : : :
---TMEILRTENIVKYFG----EFKALDGVSISVNKGDVTLIIG-PNGSGKS-TLINVIT
GFLKADEGRVYFENKDITNKEPAELYHYG-------IVRT------FQTPQPLKEMTVLE
NLLIGEICPGESPLNSLFY-KKWIPKEEEMVEKAFKILEFLKLSH-LYDRKAGELSGGQM
KLVEIGRALMTNPKMIVMDEPIAGVAPGLAHDIF-NHVLELKA-KGITFLIIEHRLDIVL
NYIDHLYVMFNGQIIAEGRGEEEIKNVLSD------PKVVEIYIGE
*
>P1;T121
sequence-c:ex7c:@:@:@:@: : : :
-------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIA
GLEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYD
NIAFP-----------LKL-RK--VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQR
QRVALGRAIVRKPQVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAM
TMGDRIAVMNRGVLQQVGSPDEVYDKPAN--------TFVAGF---
*
#end
```

With the command line:

$nest ex7.ali

there are four models to be outputted: ex7a_final.pdb, ex7b_final.pdb, ex7c_final.pdb and the composite structure ex7_final.pdb.

8) Example 8: Build models based on multiple templates

Multiple templates handling is similar to composite model building in that the specified region will be replaced by segment from other models. However, it is different from composite case because the specified region marked by dashes will try all the corresponding segments from other templates and the segment with the lowest energy is selected. For example, in the following alignment input file (ex8.ali):

```
#start
>P1;1b0u
structureX:1b0u:5:A:262:A: : : :
----NKLHVIDLHKRYG----GHEVLKGVSLQARAGDVISIIG-SSGSGKS-TFLRCINF
LEKPSEGAIIVNGQNINLVRDKDGQLKVADKNQLRLLRT-RLTMVFQHFNLWSHMTVLEN
VMEAPIQVLG-LSKHDARERALKYLAKVGIDERAQGKYPVHLSGGQQQRVSIARALAMEP
DVLLFDEPTSALDPELVGE-VLRIMQQLAEE-GKTMVVVTHEMGFARHVSSHVIFLHQGK
IEEEGDPEQVFGNPQSPRLQQFLKGSLKKLE
*
>P1;T121
sequence-a::@:@:@:@: : : :
------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAG
LEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDN
IAFPLKL-RK-VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKP
QVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGV
LQQVGSPDEVYDKPAN--------TFVAGF-
*
>P1;
composite:ex8:20:20:20:20
--------------------------aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaa----------------------aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1;1f3o
structureX:1f3o:1:A:232:A: : : :
--MIKLKNVTKTYKMGEEIIYALKNVNLNIKEGEFVSIMG-PSGSGKS-TMLNIIGCLDK
PTEGEVYIDNIKTND----------LDDDELTKIRRDKIGFVFQQFNLIPLLTALENVEL
PLIF-KYRGAMSGEERRKRALECLKMAELEERFANHKPNQLSGGQQQRVAIARALANNPP
IILADEPTGALDSKTGEKIM-QLLKKLNEEDGKTVVVVTHDINVAR-FGERIIYLKDGEV
EREEK---------------LRGF-----
*
>P1;T121
sequence-b::@:@:@:@: : : :
---VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAGLEE
PSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDNIAF
PLKL-RK---VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKPQ
```

```
VFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGVL
QQVGSPDEVYDKPAN--------TFVAGF-
*
#end

#start
>P1;1g6h
structureX:1g6h:0:A:257:A: : : :
---TMEILRTENIVKYFG----EFKALDGVSISVNKGDVTLIIG-PNGSGKS-TLINVIT
GFLKADEGRVYFENKDITNKEPAELYHYG-------IVRT------FQTPQPLKEMTVLE
NLLIGEICPGESPLNSLFY-KKWIPKEEEMVEKAFKILEFLKLSH-LYDRKAGELSGGQM
KLVEIGRALMTNPKMIVMDEPIAGVAPGLAHDIF-NHVLELKA-KGITFLIIEHRLDIVL
NYIDHLYVMFNGQIIAEGRGEEEIKNVLSD------PKVVEIYIGE
*
>P1;T121
sequence-c::@:@:@:@: : : :
-------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIA
GLEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYD
NIAFP-----------LKL-RK--VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQR
QRVALGRAIVRKPQVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAM
TMGDRIAVMNRGVLQQVGSPDEVYDKPAN--------TFVAGF---
*
#end
```

The above alignment input file contains three alignment blocks and three models will be built, i.e., model a, b and c. In the first alignment block, there is a composite pir alignment. The sequence of this composite pir alignment has two regions marked by dashes. The first dash region covers residues from 1 to 17. This 17-residue segment will be replaced by the corresponding segment from either model a,b or c whichever has the lowest energy when it is assembled on model a.

Multiple template handling is usually adopted when a query sequence identifies multiple homologous templates. Instead of relying on a particular template, the most variable region can be modeled by trying all different choices of conformation from other templates.

The above input alignment file can then be executed with the command:

$nest ex8.ali

The output will be ex8_final.pdb. Model a, b and c will not be outputted since they have no names given.

9) Example 9: fusing two domains into one structure

This is a case of composite model building. The domain can be considered as a region in one model to be replaced by the domain from the other model.

Below is an example input file of ex9.ali:

```
#start
>P1;1b0u
structureX:1b0u:5:A:262:A: : : :
----NKLHVIDLHKRYG----GHEVLKGVSLQARAGDVISIIG-SSGSGKS-TFLRCINF
LEKPSEGAIIVNGQNINLVRDKDGQLKVADKNQLRLLRT-RLTMVFQHFNLWSHMTVLEN
VMEAPIQVLG-LSKHDARERALKYLAKVGIDERAQGKYPVHLSGGQQQRVSIARALAMEP
DVLLFDEPTSALDPELVGE-VLRIMQQLAEE-GKTMVVVTHEMGFARHVSSHVIFLHQGK
IEEEGDPEQVFGNPQSPRLQQFLKGSLKKLE
*
>P1;T121
sequence-a::@:@:@:@: : : :
------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAG
LEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDN
IAFPLKL-RK-VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKP
QVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGV
LQQVGSPDEVYDKPAN--------TFVAGF-
*
>P1;
composite:ex9:20:20:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbb
#end

#start
>P1;1f3o
structureX:1f3o:1:A:232:A: : : :
--MIKLKNVTKTYKMGEEIIYALKNVNLNIKEGEFVSIMG-PSGSGKS-TMLNIIGCLDK
PTEGEVYIDNIKTND----------LDDDELTKIRRDKIGFVFQQFNLIPLLTALENVEL
PLIF-KYRGAMSGEERRKRALECLKMAELEERFANHKPNQLSGGQQQRVAIARALANNPP
IILADEPTGALDSKTGEKIM-QLLKKLNEEDGKTVVVVTHDINVAR-FGERIIYLKDGEV
EREEK----------------LRGF-----
*
>P1;T121
sequence-b::@:@:@:@: : : :
---VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAGLEE
PSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDNIAF
PLKL-RK---VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKPQ
VFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGVL
QQVGSPDEVYDKPAN--------TFVAGF-
*
#end
```

Run the program with the command:

$nest ex9.ali

10) Example 10: building multiple composite models.

Each alignment block can contain any number of composite pir alignments, and each composite pir alignment can contain any number of regions either marked as dash or with code of other models.

For example, the ex10.ali file:

```
#start
>P1;1b0u
structureX:1b0u:5:A:262:A: : : :
----NKLHVIDLHKRYG----GHEVLKGVSLQARAGDVISIIG-SSGSGKS-TFLRCINF
LEKPSEGAIIVNGQNINLVRDKDGQLKVADKNQLRLLRT-RLTMVFQHFNLWSHMTVLEN
VMEAPIQVLG-LSKHDARERALKYLAKVGIDERAQGKYPVHLSGGQQQRVSIARALAMEP
DVLLFDEPTSALDPELVGE-VLRIMQQLAEE-GKTMVVVTHEMGFARHVSSHVIFLHQGK
IEEEGDPEQVFGNPQSPRLQQFLKGSLKKLE
*
>P1;T121
sequence-a:ex10a:@:@:@:@: : : :
------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAG
LEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDN
IAFPLKL-RK-VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKP
QVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGV
LQQVGSPDEVYDKPAN--------TFVAGF-
*
>P1;
composite:ex10A:20:20
bbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
>P1;
composite:ex10B:20:20
bbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
>P1;
composite:ex10C:20:20
bbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

#end

#start
>P1;1f3o
structureX:1f3o:1:A:232:A: : : :
--MIKLKNVTKTYKMGEEIIYALKNVNLNIKEGEFVSIMG-PSGSGKS-TMLNIIGCLDK
PTEGEVYIDNIKTND----------LDDDELTKIRRDKIGFVFQQFNLIPLLTALENVEL
```

```
PLIF-KYRGAMSGEERRKRALECLKMAELEERFANHKPNQLSGGQQQRVAIARALANNPP
IILADEPTGALDSKTGEKIM-QLLKKLNEEDGKTVVVVTHDINVAR-FGERIIYLKDGEV
EREEK----------------LRGF-----
*

>P1;T121
sequence-b:ex10b:@:@:@:@: : : :
---VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIAGLEE
PSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYDNIAF
PLKL-RK---VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQRQRVALGRAIVRKPQ
VFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAMTMGDRIAVMNRGVL
QQVGSPDEVYDKPAN--------TFVAGF-
*
>P1;
composite:ex10D:20:20
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaaaaaaaaa
#end

#start
>P1;1g6h
structureX:1g6h:0:A:257:A: : : :
---TMEILRTENIVKYFG----EFKALDGVSISVNKGDVTLIIG-PNGSGKS-TLINVIT
GFLKADEGRVYFENKDITNKEPAELYHYG-------IVRT------FQTPQPLKEMTVLE
NLLIGEICPGESPLNSLFY-KKWIPKEEEMVEKAFKILEFLKLSH-LYDRKAGELSGGQM
KLVEIGRALMTNPKMIVMDEPIAGVAPGLAHDIF-NHVLELKA-KGITFLIIEHRLDIVL
NYIDHLYVMFNGQIIAEGRGEEEIKNVLSD------PKVVEIYIGE
*
>P1;T121
sequence-c:ex10c:@:@:@:@: : : :
-------VRLVDVWKVFG----EVTAVRELSLEVKDGEFMILLG-PSGCGKTTT-LRMIA
GLEEPSRGQIYIGDRLVADPEKG-IFVPPKD---------RDIAMVFQSYALYPHMTVYD
NIAFP-----------LKL-RK--VPRQEIDQRVREVAELLGLTE-LLNRKPRELSGGQR
QRVALGRAIVRKPQVFLMDEPLSNLDAKLRVR-MRAELKKLQRQLGVTTIYVTHDQVEAM
TMGDRIAVMNRGVLQQVGSPDEVYDKPAN--------TFVAGF---
*
>P1;
composite:ex10E:20:20:20:20
bbbbbbbbbbbbbbbbbbccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end
```

The above input file has three alignment blocks, each of the alignment blocks contains at least one composite pir alignment. With the multiple composite pir alignments, the user can build multiple models based on different composite choices, and selects the best model with proper energy function or other discriminatory software tools.

Run the above input file with the command:

$nest ex10.ali

You can also try other different command options.

Log File

When you run the nest program with the command:

$nest –log ex6.log ex6.ali

the log file ex6.log will be created and information during program execution will be written down to the log file. If the log file is not given, the information will be written to the screen.

Below is an example of running a command:

$nest –log ex6.log –tune 3 –fast 5 –opt 2 –out 2 ex6.ali

The ex6.log file created is as follows with bold comments in bracket:

```
reading from file:
/usr/people/xiang/jackal/library/back_small_rotamer
[this is the small backbone rotamer library]
from back_small_rotamer:the number copies for residue: A 4
from back_small_rotamer:the number copies for residue: C 7
from back_small_rotamer:the number copies for residue: D 7
from back_small_rotamer:the number copies for residue: E 6
from back_small_rotamer:the number copies for residue: F 5
from back_small_rotamer:the number copies for residue: G 11
from back_small_rotamer:the number copies for residue: H 8
from back_small_rotamer:the number copies for residue: I 4
from back_small_rotamer:the number copies for residue: K 5
from back_small_rotamer:the number copies for residue: L 5
from back_small_rotamer:the number copies for residue: M 5
from back_small_rotamer:the number copies for residue: N 8
from back_small_rotamer:the number copies for residue: P 1
from back_small_rotamer:the number copies for residue: Q 6
from back_small_rotamer:the number copies for residue: R 6
from back_small_rotamer:the number copies for residue: S 7
from back_small_rotamer:the number copies for residue: T 5
from back_small_rotamer:the number copies for residue: V 5
from back_small_rotamer:the number copies for residue: W 4
from back_small_rotamer:the number copies for residue: Y 6
[number of rotamers in backbone rotamer library]

reading from file:
/usr/people/xiang/jackal/library/side_small_rotamer

from side_small_rotamer:the number copies for residue: A 1
from side_small_rotamer:the number copies for residue: C 3
from side_small_rotamer:the number copies for residue: D 5
from side_small_rotamer:the number copies for residue: E 12
from side_small_rotamer:the number copies for residue: F 4
```

```
from side_small_rotamer:the number copies for residue: G 1
from side_small_rotamer:the number copies for residue: H 9
from side_small_rotamer:the number copies for residue: I 7
from side_small_rotamer:the number copies for residue: K 49
from side_small_rotamer:the number copies for residue: L 9
from side_small_rotamer:the number copies for residue: M 17
from side_small_rotamer:the number copies for residue: N 11
from side_small_rotamer:the number copies for residue: P 3
from side_small_rotamer:the number copies for residue: Q 19
from side_small_rotamer:the number copies for residue: R 39
from side_small_rotamer:the number copies for residue: S 3
from side_small_rotamer:the number copies for residue: T 3
from side_small_rotamer:the number copies for residue: V 3
from side_small_rotamer:the number copies for residue: W 8
from side_small_rotamer:the number copies for residue: Y 8

reading from file:
/usr/people/xiang/jackal/library/side_mix_rotamer

from side_mix_rotamer:the number copies for residue: A 1
from side_mix_rotamer:the number copies for residue: C 9
from side_mix_rotamer:the number copies for residue: D 112
from side_mix_rotamer:the number copies for residue: E 214
from side_mix_rotamer:the number copies for residue: F 55
from side_mix_rotamer:the number copies for residue: G 1
from side_mix_rotamer:the number copies for residue: H 125
from side_mix_rotamer:the number copies for residue: I 71
from side_mix_rotamer:the number copies for residue: K 401
from side_mix_rotamer:the number copies for residue: L 69
from side_mix_rotamer:the number copies for residue: M 391
from side_mix_rotamer:the number copies for residue: N 175
from side_mix_rotamer:the number copies for residue: P 9
from side_mix_rotamer:the number copies for residue: Q 262
from side_mix_rotamer:the number copies for residue: R 382
from side_mix_rotamer:the number copies for residue: S 11
from side_mix_rotamer:the number copies for residue: T 9
from side_mix_rotamer:the number copies for residue: V 7
from side_mix_rotamer:the number copies for residue: W 93
from side_mix_rotamer:the number copies for residue: Y 65

reading file:
/usr/people/xiang/jackal/library/bound_nearnext1168.dist
[This is the library for distance geometry of atom-atom
distance of a residue]

reading file:
/usr/people/xiang/jackal/library/bound_hbondind1168.dist
[This is the library for distance constraints of hydrogen
bonds]

reading pir alignment blocks...

the number of alignment blocks read: 2
```

```
change residue name into uppercase ...

remove non-standard residue name, treated as gap...

check error in alignment blocks...

the number of segments possibly to be replaced: 1
[check the composite pir alignment]

reading pdb file:1c29.pdb
[reading template structure]

Warning......
the pdb file:1c29 has breaker at:G189 P192

checking missing atoms...
the total number of missing atoms: 0
there is no missing atoms.

find corresponding residues between pdb and pir for the
structure pir:1c29

Warning......
the pdb file:1c29 has breaker at:G189 P192

checking missing atoms...
the total number of missing atoms: 0
there is no missing atoms.

find corresponding residues between pdb and pir for the
structure pir:1c29

detecting secondary structure regions using dssp
definition...

[below is the dssp secondary structure definition of the
template]
E0 -
R1 h
Y2 h
E3 h
N4 h
L5 h
F6 h
A7 h
Q8 h
L9 h
N10 h
D11 h
R12 -
R13 -
E14 -
G15 -
A16 e
```

F17 e
V18 e
P19 e
F20 e
V21 e
T22 e
L23 -
G24 -
D25 -
P26 -
G27 -
I28 h
E29 h
Q30 h
S31 h
L32 h
K33 h
I34 h
I35 h
D36 h
T37 h
L38 h
I39 h
D40 h
A41 -
G42 -
A43 -
D44 -
A45 -
L46 e
E47 e
L48 e
G49 e
V50 -
P51 -
F52 -
S53 -
D54 -
P55 -
L56 -
A57 -
D58 e
G59 -
P60 h
T61 h
I62 h
Q63 h
N64 h
A65 h
N66 h
L67 h
R68 h
A69 h
F70 h

```
A71 h
A72 -
G73 -
V74 -
T75 -
P76 h
A77 h
Q78 h
C79 h
F80 h
E81 h
M82 h
L83 h
A84 h
L85 h
I86 h
R87 h
E88 h
K89 h
H90 -
P91 -
T92 -
I93 -
P94 -
I95 e
G96 e
L97 e
L98 e
M99 e
Y100 -
A101 h
N102 h
L103 h
V104 h
F105 h
N106 -
N107 -
G108 -
I109 h
D110 h
A111 h
F112 h
Y113 h
A114 h
R115 h
C116 h
E117 h
Q118 h
V119 h
G120 -
V121 -
D122 -
S123 e
V124 e
```

L125 e
V126 e
A127 -
D128 -
V129 -
P130 -
V131 h
E132 h
E133 h
S134 -
A135 h
P136 h
F137 h
R138 h
Q139 h
A140 h
A141 h
L142 h
R143 h
H144 -
N145 -
I146 -
A147 e
P148 e
I149 e
F150 -
I151 e
C152 e
P153 -
P154 -
N155 -
A156 -
D157 -
D158 h
D159 h
L160 h
L161 h
R162 h
Q163 h
V164 h
A165 h
S166 h
Y167 h
G168 -
R169 -
G170 -
Y171 -
T172 e
Y173 e
L174 e
L175 e
S176 -
R177 -
S178 -

G179 -
V180 -
T181 e
G182 -
A183 -
E184 -
N185 -
R186 -
G187 -
P190 -
L191 -
H192 h
H193 h
L194 h
I195 h
E196 h
K197 h
L198 h
K199 h
E200 h
Y201 -
H202 -
A203 -
A204 -
P205 -
A206 e
L207 e
Q208 e
G209 e
F210 -
G211 -
I212 -
S213 -
S214 -
P215 h
E216 h
Q217 h
V218 h
S219 h
A220 h
A221 h
V222 h
R223 h
A224 -
G225 -
A226 -
A227 -
G228 e
A229 e
I230 e
S231 e
G232 -
S233 h
A234 h

I235 h
V236 h
K237 h
I238 h
I239 h
E240 h
K241 h
N242 -
L243 -
A244 -
S245 -
P246 h
K247 h
Q248 h
M249 h
L250 h
A251 h
E252 h
L253 h
R254 h
S255 h
F256 h
V257 h
S258 h
A259 h
M260 h
K261 h
A262 h
A263 -
S264 -
R265 –

[below is for the second alignment block]
detecting secondary structure regions using dssp definition...

[for the template in the secondary alignment block]
E0 -
R1 h
Y2 h
E3 h
N4 h
L5 h
F6 h
A7 h
Q8 h
L9 h
N10 h
D11 h
R12 -
R13 -
E14 -
G15 -
A16 e

F17 e
V18 e
P19 e
F20 e
V21 e
T22 e
L23 -
G24 -
D25 -
P26 -
G27 -
I28 h
E29 h
Q30 h
S31 h
L32 h
K33 h
I34 h
I35 h
D36 h
T37 h
L38 h
I39 h
D40 h
A41 -
G42 -
A43 -
D44 -
A45 -
L46 e
E47 e
L48 e
G49 e
V50 -
P51 -
F52 -
S53 -
D54 -
P55 -
L56 -
A57 -
D58 e
G59 -
P60 h
T61 h
I62 h
Q63 h
N64 h
A65 h
N66 h
L67 h
R68 h
A69 h
F70 h

```
A71 h
A72 -
G73 -
V74 -
T75 -
P76 h
A77 h
Q78 h
C79 h
F80 h
E81 h
M82 h
L83 h
A84 h
L85 h
I86 h
R87 h
E88 h
K89 h
H90 -
P91 -
T92 -
I93 -
P94 -
I95 e
G96 e
L97 e
L98 e
M99 e
Y100 -
A101 h
N102 h
L103 h
V104 h
F105 h
N106 -
N107 -
G108 -
I109 h
D110 h
A111 h
F112 h
Y113 h
A114 h
R115 h
C116 h
E117 h
Q118 h
V119 h
G120 -
V121 -
D122 -
S123 e
V124 e
```

L125 e
V126 e
A127 -
D128 -
V129 -
P130 -
V131 h
E132 h
E133 h
S134 -
A135 h
P136 h
F137 h
R138 h
Q139 h
A140 h
A141 h
L142 h
R143 h
H144 -
N145 -
I146 -
A147 e
P148 e
I149 e
F150 -
I151 e
C152 e
P153 -
P154 -
N155 -
A156 -
D157 -
D158 h
D159 h
L160 h
L161 h
R162 h
Q163 h
V164 h
A165 h
S166 h
Y167 h
G168 -
R169 -
G170 -
Y171 -
T172 e
Y173 e
L174 e
L175 e
S176 -
R177 -
S178 -

G179 -
V180 -
T181 e
G182 -
A183 -
E184 -
N185 -
R186 -
G187 -
P190 -
L191 -
H192 h
H193 h
L194 h
I195 h
E196 h
K197 h
L198 h
K199 h
E200 h
Y201 -
H202 -
A203 -
A204 -
P205 -
A206 e
L207 e
Q208 e
G209 e
F210 -
G211 -
I212 -
S213 -
S214 -
P215 h
E216 h
Q217 h
V218 h
S219 h
A220 h
A221 h
V222 h
R223 h
A224 -
G225 -
A226 -
A227 -
G228 e
A229 e
I230 e
S231 e
G232 -
S233 h
A234 h

```
I235 h
V236 h
K237 h
I238 h
I239 h
E240 h
K241 h
N242 -
L243 -
A244 -
S245 -
P246 h
K247 h
Q248 h
M249 h
L250 h
A251 h
E252 h
L253 h
R254 h
S255 h
F256 h
V257 h
S258 h
A259 h
M260 h
K261 h
A262 h
A263 -
S264 -
R265 -

check error in composite pirs...

tuning sequence alignment...

remove gaps in secondary structure...
remove zig-zag gaps...
remove gaps at terminal...
remove gaps in secondary structure...
remove zig-zag gaps...
remove gaps at terminal...

find out chain breakers and resolve...

calculate sequence conserve score at each residue between
sequence and structure pirs...

conserve score  0 E----:0
conserve score  1 R----:0
conserve score  2 Y----:0
conserve score  3 E----:0
conserve score  4 N----:0
conserve score  5 L----:0
```

```
conserve score  6 F----:0
conserve score  7 A----:0.0342041
conserve score  8 Q----:0.0746395
conserve score  9 L----:0.137302
conserve score  10 N----:0.22007
conserve score  11 D---M:0.374474
conserve score  12 R---F:0.467326
conserve score  13 R---K:0.552235
conserve score  14 E---D:0.638589
conserve score  15 G---G:0.739052
conserve score  16 A---S:0.723579
conserve score  17 F---L:0.722872
conserve score  18 V---I:0.786786
conserve score  19 P---P:0.794142
conserve score  20 F---Y:0.768504
conserve score  21 V---L:0.783214
conserve score  22 T---T:0.845495
conserve score  23 L---A:0.792081
conserve score  24 G---G:0.808659
conserve score  25 D---D:0.811852
conserve score  26 P---P:0.781514
conserve score  27 G---D:0.68458
conserve score  28 I---K:0.692889
conserve score  29 E---Q:0.668055
conserve score  30 Q---S:0.624414
conserve score  31 S---T:0.629
conserve score  32 L---L:0.664557
conserve score  33 K---N:0.616799
conserve score  34 I---F:0.62742
conserve score  35 I---L:0.645703
conserve score  36 D---L:0.630868
conserve score  37 T---A:0.573651
conserve score  38 L---L:0.612471
conserve score  39 I---D:0.593969
conserve score  40 D---E:0.564365
conserve score  41 A----:0.535275
conserve score  42 G---Y:0.610062
conserve score  43 A---A:0.683555
conserve score  44 D---G:0.707788
conserve score  45 A---A:0.789151
conserve score  46 L---I:0.849888
conserve score  47 E---E:0.915454
conserve score  48 L---L:0.928166
conserve score  49 G---G:0.962781
conserve score  50 V---I:0.954306
conserve score  51 P---P:0.982999
conserve score  52 F---F:0.970355
conserve score  53 S---S:0.969673
conserve score  54 D---D:0.966363
conserve score  55 P---P:0.965998
conserve score  56 L---I:0.908612
conserve score  57 A---A:0.925563
conserve score  58 D---D:0.924741
conserve score  59 G---G:0.911778
```

```
conserve score  60 P---K:0.834325
conserve score  61 T---T:0.865908
conserve score  62 I---I:0.840061
conserve score  63 Q---Q:0.785128
conserve score  64 N---E:0.717059
conserve score  65 A---S:0.737859
conserve score  66 N---H:0.698756
conserve score  67 L---Y:0.674739
conserve score  68 R---R:0.699723
conserve score  69 A---A:0.730964
conserve score  70 F---L:0.663549
conserve score  71 A---K:0.64078
conserve score  72 A---N:0.642643
conserve score  73 G---G:0.648573
conserve score  74 V---F:0.580493
conserve score  75 T---K:0.564937
conserve score  76 P---L:0.59296
conserve score  77 A---R:0.599471
conserve score  78 Q---E:0.598916
conserve score  79 C---A:0.60964
conserve score  80 F---F:0.653039
conserve score  81 E---W:0.609193
conserve score  82 M---I:0.603008
conserve score  83 L---V:0.617113
conserve score  84 A---K:0.601882
conserve score  85 L---E:0.570969
conserve score  86 I---F:0.55089
conserve score  87 R---R:0.582484
conserve score  88 E---R:0.52548
conserve score  89 K---H:0.49121
conserve score  90 H----:0.47149
conserve score  91 P---S:0.556046
conserve score  92 T---S:0.566009
conserve score  93 I---T:0.627151
conserve score  94 P---P:0.713786
conserve score  95 I---I:0.758094
conserve score  96 G---V:0.748515
conserve score  97 L---L:0.773174
conserve score  98 L---M:0.773711
conserve score  99 M---T:0.720578
conserve score  100 Y---Y:0.751112
conserve score  101 A---Y:0.73288
conserve score  102 N---N:0.734852
conserve score  103 L---P:0.682667
conserve score  104 V---I:0.735837
conserve score  105 F---Y:0.710469
conserve score  106 N---R:0.683812
conserve score  107 N---A:0.659185
conserve score  108 G---G:0.743946
conserve score  109 I---V:0.710049
conserve score  110 D---R:0.685552
conserve score  111 A---N:0.686504
conserve score  112 F---F:0.729071
conserve score  113 Y---L:0.668502
```

```
conserve score  114 A---A:0.672628
conserve score  115 R---E:0.618474
conserve score  116 C---A:0.632181
conserve score  117 E---K:0.637948
conserve score  118 Q---A:0.672617
conserve score  119 V---S:0.668646
conserve score  120 G---G:0.766285
conserve score  121 V---V:0.821748
conserve score  122 D---D:0.844813
conserve score  123 S---G:0.800222
conserve score  124 V---I:0.862969
conserve score  125 L---L:0.857001
conserve score  126 V---V:0.848691
conserve score  127 A---V:0.803689
conserve score  128 D---D:0.838238
conserve score  129 V---L:0.789412
conserve score  130 P---P:0.792301
conserve score  131 V---V:0.74546
conserve score  132 E---F:0.675032
conserve score  133 E---H:0.650364
conserve score  134 S---A:0.643667
conserve score  135 A---K:0.613867
conserve score  136 P---E:0.595312
conserve score  137 F---F:0.677002
conserve score  138 R---T:0.641822
conserve score  139 Q---E:0.642106
conserve score  140 A---I:0.633271
conserve score  141 A---A:0.666893
conserve score  142 L---R:0.616973
conserve score  143 R---E:0.603097
conserve score  144 H---E:0.590608
conserve score  145 N---G:0.627861
conserve score  146 I---I:0.678899
conserve score  147 A---K:0.663171
conserve score  148 P---T:0.671565
conserve score  149 I---V:0.719302
conserve score  150 F---F:0.767782
conserve score  151 I---L:0.737457
conserve score  152 C---A:0.705109
conserve score  153 P---A:0.709351
conserve score  154 P---P:0.770998
conserve score  155 N---N:0.752399
conserve score  156 A---T:0.683114
conserve score  157 D---P:0.711136
conserve score  158 D---D:0.766764
conserve score  159 D---E:0.708907
conserve score  160 L---R:0.686981
conserve score  161 L---L:0.728282
conserve score  162 R---K:0.696364
conserve score  163 Q---V:0.638496
conserve score  164 V---I:0.644532
conserve score  165 A---D:0.599972
conserve score  166 S---D:0.584323
conserve score  167 Y---M:0.586595
```

```
conserve score  168 G---T:0.594543
conserve score  169 R---T:0.625433
conserve score  170 G---G:0.720487
conserve score  171 Y---F:0.720105
conserve score  172 T---V:0.744141
conserve score  173 Y---Y:0.799011
conserve score  174 L---L:0.802468
conserve score  175 L---V:0.76481
conserve score  176 S---S:0.762412
conserve score  177 R---L:0.739563
conserve score  178 S---Y:0.737129
conserve score  179 G---G:0.788167
conserve score  180 V---T:0.76082
conserve score  181 T---T:0.782717
conserve score  182 G---G:0.790811
conserve score  183 A---A:0.766749
conserve score  184 E---R:0.624407
conserve score  185 N---E:0.550172
conserve score  186 R---E:0.484032
conserve score  187 G---I:0.416028
conserve score  188 ----P:0.326769
conserve score  189 ----K:0.328434
conserve score  190 P---T:0.429437
conserve score  191 L---A:0.485104
conserve score  192 H---Y:0.526317
conserve score  193 H---D:0.58955
conserve score  194 L---L:0.676544
conserve score  195 I---L:0.688519
conserve score  196 E---R:0.658583
conserve score  197 K---R:0.657455
conserve score  198 L---A:0.647257
conserve score  199 K---K:0.649791
conserve score  200 E---R:0.582769
conserve score  201 Y---I:0.562662
conserve score  202 H---C:0.540435
conserve score  203 A---R:0.534204
conserve score  204 A---N:0.5
conserve score  205 P---K:0.534204
conserve score  206 A---V:0.574639
conserve score  207 L---A:0.624812
conserve score  208 Q---V:0.684265
conserve score  209 G---G:0.797075
conserve score  210 F---F:0.828441
conserve score  211 G---G:0.83372
conserve score  212 I---V:0.846596
conserve score  213 S---S:0.835555
conserve score  214 S---K:0.770486
conserve score  215 P---R:0.730384
conserve score  216 E---E:0.748996
conserve score  217 Q---H:0.696828
conserve score  218 V---V:0.69545
conserve score  219 S---V:0.65528
conserve score  220 A---S:0.646113
conserve score  221 A---L:0.631899
```

```
conserve score  222 V---L:0.67481
conserve score  223 R---K:0.659479
conserve score  224 A---E:0.698306
conserve score  225 G---G:0.751237
conserve score  226 A---A:0.781514
conserve score  227 A---N:0.728996
conserve score  228 G---G:0.77681
conserve score  229 A---V:0.768286
conserve score  230 I---V:0.794965
conserve score  231 S---V:0.767152
conserve score  232 G---G:0.8588
conserve score  233 S---S:0.880428
conserve score  234 A---A:0.917012
conserve score  235 I---L:0.908612
conserve score  236 V---V:0.931794
conserve score  237 K---K:0.907115
conserve score  238 I---I:0.863057
conserve score  239 I---I:0.800117
conserve score  240 E---G:0.706417
conserve score  241 K---E:0.647686
conserve score  242 N---K:0.58824
conserve score  243 L---G:0.544239
conserve score  244 A---R:0.520577
conserve score  245 S---E:0.521606
conserve score  246 P---A:0.552524
conserve score  247 K---T:0.565267
conserve score  248 Q---E:0.598775
conserve score  249 M---F:0.600922
conserve score  250 L---L:0.642723
conserve score  251 A---K:0.59637
conserve score  252 E---K:0.585536
conserve score  253 L---K:0.562587
conserve score  254 R---V:0.554347
conserve score  255 S---E:0.52063
conserve score  256 F---E:0.530824
conserve score  257 V---L:0.502642
conserve score  258 S---L:0.452603
conserve score  259 A---G:0.401612
conserve score  260 M---I:0.340638
[Below is for the secondary alignment block]
conserve score  0 L---M:0.421879
conserve score  1 F---F:0.483115
conserve score  2 A---K:0.439208
conserve score  3 Q---D:0.407664
conserve score  4 L---G:0.361583
conserve score  5 N----:0.22702
conserve score  6 D----:0.124812
conserve score  7 R----:0.117395
conserve score  8 R----:0.118952
conserve score  9 E----:0.163008
conserve score  10 G----:0.274347
conserve score  11 A---S:0.429674
conserve score  12 F---L:0.520633
conserve score  13 V---I:0.659222
```

```
conserve score  14 P---P:0.725734
conserve score  15 F---Y:0.768504
conserve score  16 V---L:0.783214
conserve score  17 T---T:0.845495
conserve score  18 L---A:0.792081
conserve score  19 G---G:0.808659
conserve score  20 D---D:0.811852
conserve score  21 P---P:0.781514
conserve score  22 G---D:0.68458
conserve score  23 I---K:0.692889
conserve score  24 E---Q:0.668055
conserve score  25 Q---S:0.624414
conserve score  26 S---T:0.629
conserve score  27 L---L:0.664557
conserve score  28 K---N:0.616799
conserve score  29 I---F:0.62742
conserve score  30 I---L:0.645703
conserve score  31 D---L:0.630868
conserve score  32 T---A:0.573651
conserve score  33 L---L:0.612471
conserve score  34 I---D:0.593969
conserve score  35 D---E:0.564365
conserve score  36 A----:0.535275
conserve score  37 G---Y:0.610062
conserve score  38 A---A:0.683555
conserve score  39 D---G:0.707788
conserve score  40 A---A:0.789151
conserve score  41 L---I:0.849888
conserve score  42 E---E:0.915454
conserve score  43 L---L:0.928166
conserve score  44 G---G:0.962781
conserve score  45 V---I:0.954306
conserve score  46 P---P:0.982999
conserve score  47 F---F:0.970355
conserve score  48 S---S:0.969673
conserve score  49 D---D:0.966363
conserve score  50 P---P:0.965998
conserve score  51 L---I:0.908612
conserve score  52 A---A:0.925563
conserve score  53 D---D:0.924741
conserve score  54 G---G:0.911778
conserve score  55 P---K:0.834325
conserve score  56 T---T:0.865908
conserve score  57 I---I:0.840061
conserve score  58 Q---Q:0.785128
conserve score  59 N---E:0.717059
conserve score  60 A---S:0.737859
conserve score  61 N---H:0.698756
conserve score  62 L---Y:0.674739
conserve score  63 R---R:0.699723
conserve score  64 A---A:0.730964
conserve score  65 F---L:0.663549
conserve score  66 A---K:0.64078
conserve score  67 A---N:0.642643
```

```
conserve score  68 G---G:0.648573
conserve score  69 V---F:0.580493
conserve score  70 T---K:0.564937
conserve score  71 P---L:0.59296
conserve score  72 A---R:0.599471
conserve score  73 Q---E:0.598916
conserve score  74 C---A:0.60964
conserve score  75 F---F:0.653039
conserve score  76 E---W:0.609193
conserve score  77 M---I:0.603008
conserve score  78 L---V:0.617113
conserve score  79 A---K:0.601882
conserve score  80 L---E:0.570969
conserve score  81 I---F:0.585095
conserve score  82 R---R:0.622919
conserve score  83 E---R:0.602208
conserve score  84 K---H:0.556404
conserve score  85 H---S:0.584377
conserve score  86 P---S:0.626675
conserve score  87 T---T:0.636972
conserve score  88 I----:0.606016
conserve score  89 P---P:0.71894
conserve score  90 I---I:0.739315
conserve score  91 G---V:0.734636
conserve score  92 L---L:0.73897
conserve score  93 L---M:0.773711
conserve score  94 M---T:0.720578
conserve score  95 Y---Y:0.751112
conserve score  96 A---Y:0.73288
conserve score  97 N---N:0.734852
conserve score  98 L---P:0.682667
conserve score  99 V---I:0.735837
conserve score  100 F---Y:0.710469
conserve score  101 N---R:0.683812
conserve score  102 N---A:0.659185
conserve score  103 G---G:0.743946
conserve score  104 I---V:0.710049
conserve score  105 D---R:0.685552
conserve score  106 A---N:0.686504
conserve score  107 F---F:0.729071
conserve score  108 Y---L:0.668502
conserve score  109 A---A:0.672628
conserve score  110 R---E:0.618474
conserve score  111 C---A:0.632181
conserve score  112 E---K:0.637948
conserve score  113 Q---A:0.672617
conserve score  114 V---S:0.668646
conserve score  115 G---G:0.766285
conserve score  116 V---V:0.821748
conserve score  117 D---D:0.844813
conserve score  118 S---G:0.800222
conserve score  119 V---I:0.862969
conserve score  120 L---L:0.857001
conserve score  121 V---V:0.848691
```

```
conserve score  122 A---V:0.803689
conserve score  123 D---D:0.838238
conserve score  124 V---L:0.789412
conserve score  125 P---P:0.792301
conserve score  126 V---V:0.74546
conserve score  127 E---F:0.675032
conserve score  128 E---H:0.650364
conserve score  129 S---A:0.643667
conserve score  130 A---K:0.613867
conserve score  131 P---E:0.595312
conserve score  132 F---F:0.677002
conserve score  133 R---T:0.641822
conserve score  134 Q---E:0.642106
conserve score  135 A---I:0.633271
conserve score  136 A---A:0.666893
conserve score  137 L---R:0.616973
conserve score  138 R---E:0.603097
conserve score  139 H---E:0.590608
conserve score  140 N---G:0.627861
conserve score  141 I---I:0.678899
conserve score  142 A---K:0.663171
conserve score  143 P---T:0.671565
conserve score  144 I---V:0.719302
conserve score  145 F---F:0.767782
conserve score  146 I---L:0.737457
conserve score  147 C---A:0.705109
conserve score  148 P---A:0.709351
conserve score  149 P---P:0.770998
conserve score  150 N---N:0.752399
conserve score  151 A---T:0.683114
conserve score  152 D---P:0.711136
conserve score  153 D---D:0.766764
conserve score  154 D---E:0.708907
conserve score  155 L---R:0.686981
conserve score  156 L---L:0.728282
conserve score  157 R---K:0.696364
conserve score  158 Q---V:0.638496
conserve score  159 V---I:0.644532
conserve score  160 A---D:0.599972
conserve score  161 S---D:0.584323
conserve score  162 Y---M:0.586595
conserve score  163 G---T:0.594543
conserve score  164 R---T:0.625433
conserve score  165 G---G:0.720487
conserve score  166 Y---F:0.720105
conserve score  167 T---V:0.744141
conserve score  168 Y---Y:0.799011
conserve score  169 L---L:0.802468
conserve score  170 L---V:0.76481
conserve score  171 S---S:0.762412
conserve score  172 R---L:0.739563
conserve score  173 S---Y:0.737129
conserve score  174 G---G:0.788167
conserve score  175 V---T:0.76082
```

```
conserve score  176 T---T:0.782717
conserve score  177 G---G:0.790811
conserve score  178 A---A:0.766749
conserve score  179 E---R:0.624407
conserve score  180 N---E:0.550172
conserve score  181 R---E:0.484032
conserve score  182 G---I:0.416028
conserve score  183 ----P:0.326769
conserve score  184 ----K:0.328434
conserve score  185 P---T:0.429437
conserve score  186 L---A:0.485104
conserve score  187 H---Y:0.526317
conserve score  188 H---D:0.58955
conserve score  189 L---L:0.676544
conserve score  190 I---L:0.688519
conserve score  191 E---R:0.658583
conserve score  192 K---R:0.657455
conserve score  193 L---A:0.647257
conserve score  194 K---K:0.649791
conserve score  195 E---R:0.582769
conserve score  196 Y---I:0.562662
conserve score  197 H---C:0.540435
conserve score  198 A---R:0.534204
conserve score  199 A---N:0.5
conserve score  200 P---K:0.534204
conserve score  201 A---V:0.574639
conserve score  202 L---A:0.624812
conserve score  203 Q---V:0.684265
conserve score  204 G---G:0.797075
conserve score  205 F---F:0.828441
conserve score  206 G---G:0.83372
conserve score  207 I---V:0.846596
conserve score  208 S---S:0.835555
conserve score  209 S---K:0.770486
conserve score  210 P---R:0.730384
conserve score  211 E---E:0.748996
conserve score  212 Q---H:0.696828
conserve score  213 V---V:0.69545
conserve score  214 S---V:0.65528
conserve score  215 A---S:0.646113
conserve score  216 A---L:0.631899
conserve score  217 V---L:0.67481
conserve score  218 R---K:0.659479
conserve score  219 A---E:0.698306
conserve score  220 G---G:0.751237
conserve score  221 A---A:0.781514
conserve score  222 A---N:0.728996
conserve score  223 G---G:0.77681
conserve score  224 A---V:0.768286
conserve score  225 I---V:0.794965
conserve score  226 S---V:0.767152
conserve score  227 G---G:0.8588
conserve score  228 S---S:0.880428
conserve score  229 A---A:0.917012
```

```
conserve score  230 I---L:0.908612
conserve score  231 V---V:0.931794
conserve score  232 K---K:0.907115
conserve score  233 I---I:0.863057
conserve score  234 I---I:0.765912
conserve score  235 E---G:0.631777
conserve score  236 K---E:0.522874
conserve score  237 N---K:0.429629
conserve score  238 L----:0.324183
conserve score  239 A----:0.284965
conserve score  240 S----:0.307381
conserve score  241 P---G:0.38454
conserve score  242 K---R:0.441781
conserve score  243 Q---E:0.491953
conserve score  244 M---A:0.543152
conserve score  245 L---T:0.58601
conserve score  246 A---E:0.592192
conserve score  247 E---F:0.598814
conserve score  248 L---L:0.673404
conserve score  249 R---K:0.650201
conserve score  250 S---K:0.625176
conserve score  251 F---K:0.640723
conserve score  252 V---V:0.679845
conserve score  253 S---E:0.600976
conserve score  254 A---E:0.577935
conserve score  255 M---L:0.552922
conserve score  256 K---L:0.487327
conserve score  257 A---G:0.395671
conserve score  258 A---I:0.323692

write down the alignment blocks actually used in the
program to: ex6.ali~

[model building for the first alignment block.]
model building for the 0th alignment blocks

finish initial model building...
optimizing initial model of the 0th alignment blocks...
model building for the 1th alignment blocks
finish initial model building...

optimizing initial model of the 1th alignment blocks...
[composite model building]
handling blocks for composite pirs...

working on composite model building :ex6
trying to replace segments: M0---G13
with segment from model with id:b

superimpose the two segments with corresponding residues...
[corresponding residues: residue in model a-residue in
model b]
corresponding residues: D14---D14
corresponding residues: P15---P15
```

```
corresponding residues: D16---D16
corresponding residues: K17---K17
corresponding residues: Q18---Q18
corresponding residues: S19---S19
corresponding residues: T20---T20
corresponding residues: L21---L21
corresponding residues: N22---N22
corresponding residues: F23---F23
corresponding residues: L24---L24
corresponding residues: L25---L25
corresponding residues: A26---A26
corresponding residues: L27---L27
corresponding residues: D28---D28
corresponding residues: E29---E29
corresponding residues: Y30---Y30
corresponding residues: A31---A31
corresponding residues: G32---G32
corresponding residues: A33---A33
corresponding residues: I34---I34

the above rmsd :0.0118114 A

output intermediate composite structures:
ex6_composite_tmp.1.pdb

[output intermediate composite structure]
output the composite structure: ex6_composite.2.pdb

refinement models...
the model with code name:ex6

refinement loop regions with insertion and deletions..

refine regions between: D41---Y44
refine regions between: R89---S94
write down intermediate structures in refinement of loop
regions: ex6_loop_refine_tmp.3.pdb
write down intermediate structures in refinement of loop
regions: ex6_loop_refine_tmp.4.pdb
write down intermediate structures in refinement of loop
regions: ex6_loop_refine_tmp.5.pdb
refine regions between: E188---T192
write down intermediate structures in refinement of loop
regions: ex6_loop_refine_tmp.6.pdb
write down intermediate structures in refinement of loop
regions: ex6_loop_refine_tmp.7.pdb
minimize regions between: D41---Y44
minimize regions between: R89---S94
minimize regions between: E188---T192

output structure after loop
refinement...ex6_loop_refine.8.pdb
[intermdiate refinement composite structure]
```

```
refinement sidechains with scap...
with original conformation set as initial conformations for
sidechain refinement...

calculate solvent accessible surface....
building multiple rotamers for each side-chain...
building random initial conformation...

output structure after sidechain refinement...
ex6_side_refine.9.pdb

write out the final models...

output the final model: ex6_final.pdb
[This is final model we need]
```

5.2) scap

Introduction
Commands
Tutorial
Log File

Introduction

a) What is Scap?

Scap is a program for protein side-chain prediction and residue mutation. The program can make prediction on all residues or on certain residues in a protein of multiple chains. It can automatically detect if the residue to be predicted or mutated is backbone only, or complete with all side-chain atoms. If the residue is backbone only, it will first add side-chains and then do predictions. In the process of construction of side-chains to the backbone, positions of atom CB are averaged by assembling all rotamers to the backbone; if the residue is to be mutated, the residue is first mutated accordingly and prediction is then performed;

The scap performs side-chain prediction using coordinate rotamer libraries. There are four coordinate rotamer libraries included in the library: side_small_rotamer, side_medium_rotamer, side_large_rotamer and side_mix_rotamer. The side_small_rotamer library has 214 side-chain rotamers with 40-degree chi angle cutoff. This is equivalent to the 214 sidechain rotamer library of Tuffery's with standard bond angle and length. The side_medium_rotamer library has 3222 side-chain rotamers library compiled from 297 chains with 20 degree cutoff and 96% representation; the side_large_rotamer library has 6487 side-chain rotamers compiled from 297 chains with 10 degree cutoff and 96% representation; the side-mix-rotamer library is a mix of the side_medium_rotamer and side_large_rotamer libraries. It includes all rotamers from side_large_rotamer except for ARG, LYS and MET which come from the side_medium_rotamer library.

The four coordinate rotamer and dihedral angle rotamer libraries can be downloaded from website:

http://trantor.bioc.columbia.edu/~xiang/sidechain/index.html

More detailed information about the methods and algorithms in scap can be found in the paper:

Xiang Z, Honig B. Extending the accuracy limits of prediction for side-chain conformations. J. Mol. Biol. 2001 311:421-30.

Xiang,Z; Honig,B. Colony energy improves prediction of exposed side-chain conformations. (to be submitted)

Commands

For all the programs included in JACKAL package, you can find their usage by running the program with no arguments.

After unpacking the jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/scap. This directory contains all examples used in this tutorial for running scap program. Go to this directory and type:
$scap
scap will print out message about its usage as following:

****
scap is a protein side-chain program with the following capabilities:
A. predict side-chain conformations of a whole protein
B. predict specified side-chains in a protein
C. mutate specified residues in a protein
questions? please refer to Dr.Jason Z. Xiang at zx11@columbia.edu
****
Usage: scap -prm num -min num -out num -ini num -self num -seed num file.pdb
file_scap.list
-prm force parameter flag. default is 1
-prm 1: CHARMM22 with all atom model
-prm 2: AMBER with all atom model
-prm 3: CHARMM with heavy atom model
-prm 4: AMBER with heavy atom model
-min side-chain refinement with 2 degree steps. default is 0.
num ranges from 0-4 with 4 most thorough refinement.
-seed random seed number
-out output model. num from 0-2. default is 0.
-out 0: only the final structure outputted.
-out 1: output the final and intermediate structure.
-out 2: output all residue conformations of different rotamers plus -out 1
-ini the number of initial structures tried. default is 3.
-self option to retain original sidechain as rotamer.default is 0
-self 0: original sidechain conformations not used as rotamer.
-self 1: original sidechain conformations used as a regular rotamer
-self 2: original sidechain conformations used as rotamer and used as the first initial conformations.
-rtm side-chain rotamer library. default is 1.
-rtm 1: large side-chain rotamer library
-rtm 2: mix side-chain rotamer library
-rtm 3: medium side-chain rotamer library
-rtm 4: small side-chain rotamer library
file.pdb pdb file
file_scap.list residue list whose sidechain is to be predicted or mutated.optional. default is none the residuelist must be ended with _scap.list

When you run scap, you type: scap plus option and plus the integer value for the option and plus the pdb file. If there is something wrong with your command, scap will exit and print out an error message. The above options can be omitted, where scap will use their default values; or several options can be used at the same time.

prm option

The option "-prm num" is used to decide which force field is used, either CHARMM or AMBER and which atom model is used, either all atom model or heavy atom model. With the option 1 or 3, CHARMM22 force field will be used, i.e., the torsion energy, van der Waals radius and charge parameters takes from CHARMM. In the case of all atom model, missing protons will first be added back to each residue in the protein and scap then is executed on the protein with all hydrogens assembled. In the case of the heavy atom model, all protons on the protein will first be stripped and scap is then executed. In the heavy atom model, charges of the hydrogens will give back to its parent heavy atom so that the functional group still keep the same amount of total charges.

ini option

Since side-chain prediction is a combinatorial problem, the more conformation sampled, the more chance lower-energy conformation located. One strategy is to use as many as possible different initial conformations, and each of these initial conformations is sampled in the rotamer library independently. ini option is used to decide how many initial conformations should be tried. In the case of "-ini 1", i.e., only one initial conformation is tried, the initial conformation will be constructed in the way that the side-chains has the lowest interaction energy with the backbone. If there are more than 1 initial conformation tried, the first one is still the one with the lowest interaction energy with the backbone and the other conformations are constructed based on the lowest energy conformation ever achieved previously by randomizing 30% of its side-chains. However in the case of "-self 2", the first initial conformation will be the one reading from the pdb file and the second initial conformation is the one with the lowest interaction energy with the backbone.

min option

The option "-min num" is used to minimize side-chain conformations. The minimization is performed by sampling more conformational space around the side-chain rotamer library. This option may be activated because the rotamer library may not be detailed enough to cover all possible side-chain conformations. Even in the case of the large rotamer library which has 7000 rotamers, the rotamer library can only cover 96% of the possible side-chain conformations with 10 degree cutoff. With the option "-min 1", the scap program will first iteratively sample all side-chain rotamers until convergence, the final lowest-energy conformation will then be minimized by refining the side-chain conformation with 2 degrees rotation on each bond to search for lower energy conformations around the rotamer. With the option "-min 2", the refinement of the side-chain conformation will be applied to all the conformations from different initial conformations. In the case of option 3 or 4, the minimization is more thorough in that the

refined rotamers will be added back to the rotamer library. Only option 1 or 2 is recommended because if the user wants to explore more conformation space, the alternative approach exists by using a larger number of initial conformations.

seed option
The seed option is just to give a random seed number to the scap program. The default is 18120.

out option
The out option is used to direct the scap program how to output final structures. With option 0, only the final structure is outputted, i.e., the one with the lowest energy among all the different initial conformations; with option 1, all the structures from the different initial conformations are outputted; with option 2, all possible rotamer conformations of residues will be outputted. This option is useful when the user wants to investigate different possible side-chain conformations available for each residue. The user may also apply his/her own energy to sort out the best rotamers. The option 2 provides a chance for the user to write his own simple side-chain prediction program without investing much time taking care of rotamers since all possible side-chain conformations of a residue have already been saved to the disk. Options are additive, i.e., option 2 includes option 1. The final prediction will have a name like file_scap.pdb, where file is the original input pdb file name.

self option
The self option is used to include the original side-chain conformation either as the starting initial conformation or as a rotamer. With ini option 0, the original side-chain conformation is discarded when building the initial conformations. With the option 1, the original side-chain conformation will be treated as a regular rotamer and added to the rotamer library and then the original side-chains from the protein are stripped down; with option 2, the original side-chain conformation is used as the first initial conformations plus option 1. If the user feels that the input pdb file contains reasonable side-chain conformations and he does not want to predict the side-chain conformations totally from scratch, this option should be activated.

rtm option
This rtm option is used to decide which rotamer library should be used. With option 1, the side_large_rotamer library will be used; with option 2, the side_mix_rotamer library is used; with option 3, the side_medium_rotamer library is used; with option 4, the side_small_rotamer library is used. Generally speaking, the more detailed rotamer library the more accurate results with more cpu time cost. However, the user can create his own rotamer library and put it into the library directory. Suppose the user has his own library called “side_x”, he can change the name of “side_x” to one of the four rotamer names provided above and he fools the scap program to use the new rotamer library. Since scap does not support dihedral angle rotamer library anymore, the dihedral angle rotamer library can be converted into a coordinate rotamer library. In coordinate rotamer library, each side-chain rotamer is actually a standalone residue conformation with the side-chain chi angles equivalent to that specified by the dihedral angle rotamer library. Each

coordinate rotamer library has three lines following it. The three lines starts with “HEAD 1”, “HEAD 2” and “HEAD 3”. These three head lines are actually the atomic positions of CA,C and O of a residue ahead of the rotamer with the peptide covalent bond.

file_scap.list

This file is used when the user wants to do side-chain prediction/mutation on a number of specified residues. When this file does not exist, the scap program will try to make side-chain predictions on all residues in the input pdb file. With this file available, which must end with **_scap.list** and should be at the argument position just following the input pdb file, the scap program will first read this scap list file and find out which residue is to be predicted or mutated. The scap list file must have the format as follows:

A,1,K → meaning: mutate the residue with id number=1 in chain with id A to lysine
1,P → meaning: mutate the residue with id number=1 in chain with id ' ' to proline
2 → meaning: predict side-chain conformation in residue with residue id number=2 in chain with id ' '.
A,1 → meaning: predict side-chain conformation of residue with id =1 in Chain ‘A’.

In scap list file, the line starting with “!” is comment line and thus ignored. Each valid line should contain no more than three characters separated by no more than two commas.

When there are two commas in the line, the three characters separated by them will be treated sequentially as chain id, the residue id and the new name of the residue mutated. If the first character is missing, it means that the chain id is an empty space. For example:

,15,K

which means mutating residue with id 15 to K of the chain with empty id ‘ ’. However, if the third character is missing, it means the residue will only be predicted:

,18,

which means prediction of the residue with id 18 in the chain with id equal to empty space.

If the second character is missing, scap will exit with error message.

When there are one comma in the line, the two character separated by the comma will either be chain id +residue id or residue id+residue name. If the second character is a numerical number from 0 to 9, then the line means chain id+residue id, where only side-chain prediction is involved; otherwise, the line means residue id+residue name, where the default chain is the one with empty id and side-chain mutation is involved if the residue name supplied is a standard amino acid name. For example:

100,K

Since the second character K is not a numeral number, it means residue id is 100 and new residue name is K. One more example:

145,

where the second character does not exist, it means empty. So the line means residue id is 145 and the new residue name is empty space, which is not a standard name, so keep the original name thus the residue with id 145 will be predicted rather than mutated.

If there is no comma in the line, it means that the prediction of the residue with id provided in the chain with empty character. For example,

120

which means prediction on residue with id 120.

Tutorial

This tutorial can be found in the directory jackal/scap

1) **Example 1: side-chain prediction on all residues in ex1.pdb**

With this command:

$scap –prm 1 –ini 3 ex1.pdb

The above command means side-chain prediction with the all atom model using the force field CHARMM 22 together with 3 initial conformations. The output should be ex1_scap.pdb.

You can also use other option choices:

$scap –prm 3 –min 1 -ini 3 –rtm 1 ex1.pdb

which means using the large side-chain rotamer library and the CHARMM heavy-atom model. The predicted structure will be subjected to further refinement before written down as ex1_scap.pdb.

or

$scap –prm 3 –min 1 -ini 1 –rtm 1 –self 2 –out 2 ex1.pdb

which means the initial conformation is the original pdb file and the original side-chain conformation is treated as a regular rotamer. Besides the regular output ex1_scap.pdb, the other output ex1_scap_rtm.pdb includes all other possible side-chain conformations.

Generally speaking, scap is more accurate in the case of the all-atom model and with large value of –ini. However the more initial conformations involved, the more CPU cost. For the best performance and efficiency, I RECOMMEND:

$scap –prm 1 –ini 3 –rtm 1 –self 2 –m 1 ex1.pdb

The fastest execution of scap is with this command:

$scap –prm 3 –min 1 -ini 3 –rtm 1 ex1.pdb

2) Example 2: Side-chain mutation on a number of specific residues.

Example 2 is the case where side-chain mutation is required on a number of specific residues. This can be achieved by using the scap list file as discussed in the command section. For example,

$scap –prm 3 –min 1 -ini 3 –rtm 1 ex2.pdb ex2_scap.list

File ex2_scap.list is as follows:

A,5,P
A,6,L
B,147,L

The above lines means mutating residue 5 to Pro and residue 6 to Leu in chain A, mutating residue 147 to Leu in chain B. The three mutated side-chains are then predicted with heavy-atom models with the CHARMM22 force field using 3 initial conformations and the large side-chain rotamer library. The output will be ex2_scap.pdb.

3) Example 3: Side-chian prediction and mutation on certain residues

Example 3 is the case where side-chain mutation and prediction are required on a number of specific residues. This can be achieved by using the scap list file as discussed in the command section. For example,

$scap –prm 3 –min 1 -ini 3 –rtm 1 ex3.pdb ex3_scap.list

The file ex3_scap.list is as follows:

A,5,P
A,6,L
B,147,L
B,140
A,1
A,23

This above lines means mutating residue 5 to Pro and residue 6 to Leu in chain A, mutating residue 147 to Leu in chain B. The three mutated side-chains plus residue 1 and 23 in chain A and 140 in chain B are then predicted with heavy-atom models with the CHARMM22 force field using 3 initial conformations and the large side-chain rotamer library. The output will be ex2_scap.pdb.

Log File

When you run the scap program with the command:

$scap –prm 3 –min 1 -ini 1 –rtm 2 –out 2 ex3.pdb ex3_scap.list

the log file shown on the screen will display information during program execution .

The log file is as follows with comments bold in bracket:

```
reading from file: ../library/back_small_rotamer
[reading backbone rotamer library]
from back_small_rotamer:the number copies for residue: A 4
from back_small_rotamer:the number copies for residue: C 7
from back_small_rotamer:the number copies for residue: D 7
from back_small_rotamer:the number copies for residue: E 6
from back_small_rotamer:the number copies for residue: F 5
from back_small_rotamer:the number copies for residue: G 11
from back_small_rotamer:the number copies for residue: H 8
from back_small_rotamer:the number copies for residue: I 4
from back_small_rotamer:the number copies for residue: K 5
from back_small_rotamer:the number copies for residue: L 5
from back_small_rotamer:the number copies for residue: M 5
from back_small_rotamer:the number copies for residue: N 8
from back_small_rotamer:the number copies for residue: P 1
from back_small_rotamer:the number copies for residue: Q 6
from back_small_rotamer:the number copies for residue: R 6
from back_small_rotamer:the number copies for residue: S 7
from back_small_rotamer:the number copies for residue: T 5
from back_small_rotamer:the number copies for residue: V 5
from back_small_rotamer:the number copies for residue: W 4
from back_small_rotamer:the number copies for residue: Y 6
[number of rotamers in backbone rotamer library]


reading from file: ../library/side_large_rotamer

from side_large_rotamer:the number copies for residue:A 1
from side_large_rotamer:the number copies for residue:C 9
from side_large_rotamer:the number copies for residue:D 112
from side_large_rotamer:the number copies for residue:E 821
from side_large_rotamer:the number copies for residue: F 55
from side_large_rotamer:the number copies for residue: G 1
from side_large_rotamer:the number copies for residue:H 125
from side_large_rotamer:the number copies for residue: I 71
from side_large_rotamer:thenumber copies for residue:K 1948
```

```
from side_large_rotamer:the number copies for residue: L 69
from side_large_rotamer:the number copies for residue:M 391
from side_large_rotamer:the number copies for residue:N 175
from side_large_rotamer: the number copies for residue:P 9
from side_large_rotamer:the number copies for residue:Q 785
from side_large_rotamer:the number copies forresidue:R 1730
from side_large_rotamer:the number copies for residue: S 11
from side_large_rotamer the number copies for residue: T 9
from side_large_rotamer: the number copies for residue: V 7
from side_large_rotamer:the number copies for residue: W 93
from side_large_rotamer:the number copies for residue: Y 65
[number of rotamers in side-chain rotamer library]

reading pdb file:ex3.pdb

****residue :5A  in chain A will be mutated to:P
****residue :6V  in chain A will be mutated to:K
****residue :147G  in chain B will be mutated to:L
****residue :140F  in chain B will be predicted***
Warning! residue: A,1
does not exist!
****residue :23I  in chain A will be predicted***
[reporting information of scap list]

calculate solvent accessible surface....
building multiple rotamers for each side-chain...
building random initial conformation...
write   down   scap   prediction   for   diffeent   initial
conformations: ex3_scap.0.pdb
write   down   scap   prediction   for   diffeent   initial
conformations:ex3_scap.1.pdb
minimized energy:-26.683
write down the final scap prediction: ex3_scap.pdb
```

5.3) loopy

Introduction
Commands
Tutorial
Log File

Introduction

a) What is Loopy?

Loopy is a program for protein loop prediction, segment mutation, adding missing residues, residue insertion, and residue deletion. The loopy program uses colony energy approach to take account entropy effect and uses fast torsion space minimizer to speed up the conformation sampling.

b) Prediction of loop conformation.

If the segment to be predicted already exists, loopy will delete the original segment and predict a new one assembled onto the stems. The prediction is performed in 5 steps: a) generating a large number of random conformations using ab-initio method and closing each of the candidates with random tweak method; b) minimizing each of the closed random candidates with our fast torsion space minimizer; c) doing side-chain predictions on each of the candidates; d) using colony energy to select the best candidates; e) using loop fusion to sample more conformation space; f) repeat d)-e) until convergence.

For detailed information about methods and algorithms, please read the paper:

Xiang, Z., Soto, C and Honig, B. Evaluating configurational free energies: the colony energy concept and its application to the problem of protein loop prediction. 2002. *Proc. Natl. Acad. Sci. USA* 99:7432-7437.

Xiang,Z; Honig,B. Fast energy minimization in torsion space with end constraints. *(to be submitted)*.

c) Segment Mutation

If the segment is to be mutated, you must provide new residue names of the segment. Loopy will first delete the original segment and assemble a new segment with residue names provided. Loopy will consider it a normal loop prediction with the new segment assembled. Segment mutation can also be done using our side-chain prediction program (Scap). The difference is that in Scap, the backbone is fixed while in loopy the backbone is to be predicted.

d) Residue Insertion and Deletion

Loopy handles residue insertion and deletion by first deleting the original segment and generating a new segment with the sequence provided. If the sequence provided has fewer number of residues, then the new segment created is equivalent to residue deletion, otherwise it is residue insertion if the number of sequence provided has more residues than the original segment. After the new segment is generated, loopy will predict a new segment conformation as in case b).

e) Add Missing Residues

If the segment to be predicted has residue missing, loopy can first generate a new segment complete with all residues to replace the original one and perform conformation sampling and sorting with algorithms in case b).

Commands

For all the programs included in JACKAL package, you can find their usage by running the program with no arguments.

After unpacking jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/loopy. This directory contains all examples used in this tutorial for running loopy program. Go to this directory and type:
$loopy
loopy will print out message about its usage as following:

****
loopy is a protein segment conformation prediction with the following capabilities:
A. predict loop conformation
B. segment mutation
C. insertion and deletion
questions? please refer to Dr. Jason Z. Xiang at zx11@columbia.edu
****
loopy -prm num -obj num-num -seed num -ini num -cid char -out num -sec char file.pdb
-prm force parameters. default 1
-prm 1: CHARMM22 with all atom model
-prm 2: AMBER with all atom model
-prm 3: CHARMM with heavy atom model
-prm 4: AMBER with heavy atom model
-obj num-num: start and end id of loops to be predicted
-seed num: random seed number
-ini: number of initial conformations generated.default is 100
-cid: chain id. default is the chain with empty space
-out: number of outputs. default is 1
-res: sequence of the segment. optional.
-fast: fast mode. default is 0.
-fast 0: fast mode off. energy minimization on candidate conformations
-fast 1: fast mode on. no energy minimization on candidate conformations
file.pdb pdb file

When you run loopy, you type: loopy plus option plus the integer value for the option plus the pdb file. If there is something wrong with your command, loopy will exit and print out an error message. The above options can be omitted, where loopy uses their default values; or several options can be used at the same time.

prm option
The option "-prm num" is used to decide which force field is used, either CHARMM or AMBER, and which atom model is used, either all atom model or heavy atom model.

With option 1 or 3, CHARMM22 force field will be used, i.e., torsion energy, van der Waals radius and charge parameters taken from CHARMM. In the case of all atom model, missing protons will first be added back to each and loopy is then executed on the protein with all hydrogen atoms assembled. In the case of heavy atom model, all protons on the protein will first be stripped and loopy is then executed. In the heavy atom model, charge on hydrogen atoms will give back to their parent heavy atoms so that functional groups still keep the total charges unchanged.

ini option
Since loop prediction is a combinatorial problem, the more conformation sampled, the more chance lower-energy conformation obtained. One strategy is to use as many as possible different initial random conformations, and each of these initial conformations is sampled in the backbone rotamer library independently. ini option is used to decide how many initial conformation should be tried. In the case of "-ini 100", i.e., 100 initial conformation candidates are generated, these initial conformations will be closed with random tweak method.

seed option
Seed option is just to give a random seed number to loopy program. The default is 18120.

obj option
obj option is used to specify the start and end residue ids of the loop segment. It should be used in a way such as "-obj 3-10", meaning the segment from residue 3 to 10. However, if the residue id is minus, it should be used with caution because loopy cannot distinguish minus and dash. So in that case, it is better to re-index the original pdb file so that the start and end residue ids of the segment have non-negative value.

out option
Out option is used to direct loopy program how many conformations are outputted. The number of initial conformations should be larger than the number of outputs; otherwise, Loopy will try to increase the number of initial conformations. Suppose in the case of option "-ini 1000 –out 3", loopy will first generate 1000 closed loop candidates and keep only the first 100 of the lowest energy. These 100 loop candidates are then sorted with colony energy. The first 1-33 candidates are then subjected to recombination to produce more candidates around them, the lowest energy is written down in file "file_loopy.1.pdb", where the file is the pdb name; the candidates from 34-66 go to the same procedure as above and the second prediction is written down; the third output is out of the candidates from 67-100. The reason that loopy does not directly take the first three among the 100 candidates as the output is that with colony energy approach, the first several candidates will usually have similar conformations.

However, user could choose to output every candidate conformation by setting the number of output equal to that of the number of candidates.

res option

res option is used to provide loopy with the sequence of a segment. Suppose a segment is from 10-17 and the string sequence after option –res is "AGHGGGKLLL", the original segment of 8 residues will be replaced with a new segment of 10 residues with the sequence specified by –res option. If the number of residues in the new sequence is more than that in the original segment, it is equivalent to residue insertion; if less, it is equivalent to residue deletion; if equal, it is equivalent to segment mutation . If there are residues missing in the original segment, the missing residues can be added back by providing loopy with a new string of residues with missing residues included.

cid option

cid option is used to specify the chain id of the segment. If the chain id is an empty space, this option should not be used because it happens to be default value.

fast option

fast option is used to activate the fast mode of loopy. If the fast mode is on, then the initial candidates generated will not be minimized in torsion angle space.

Tutorial

The tutorial of loopy can be found in the directory jackal/loopy

1) **Example 1: prediction of loop conformation**
With this command:

$ loopy –obj 17-23 -prm 3 -out 1 –ini 200 ex1.pdb

The above command line means predicting the segment conformation from residue 17 to 23 in the default chain (chain id with empty space ‘ ’) using CHAARMM22 force field and heavy atom model. 200 of closed loop candidates are generated randomly. Only one output is required.

The final output is ex1_loopy.1.pdb.

2) Example 2: prediction of loop conformation in multi-chain proteins

$ loopy –obj 43-49 -prm 3 -out 1 –ini 200 –cid C ex2.pdb

The above command means predicting the loop from 43-49 in chain ‘C’ in ex2.pdb. The output will be ex2_loopy.pdb.

3) Example 3: add missing residues

Loopy can add missing residues to a segment. The philosophy is just as in the case of 4) and 5). In ex3.pdb, the residue 19A and 20C are missing.

The following command tries to replace the gapped segment with a new complete segment:

$ loopy –obj 17-23 -prm 3 -out 1 –ini 200 –res LSACKRL ex3.pdb

The sequence “LSACKRL” is the complete sequence of the segment from 17 to 23. The new segment with sequence specified by –res option is then predicted with ab-initio from scrach.

4) Example 4: Segment mutation
Example 4 is the case where the original segment is mutated and then the conformation of the new segment is predicted from scratch. Though scap can also mutate residues, the backbone of segment does not change in scap.

$ loopy –obj 17-23 -prm 3 -out 1 –ini 200 –res AAAAAAA ex4.pdb

The original segment of 7 residues is now mutated to “AAAAAAA”, and the new segment is then predicted.

Example 5: Residue Deletion/Insertion
Example 5 is the case where the original segment is replaced with a new segment of different number of residues. The conformation of the new segment is then predicted:

$ loopy –obj 17-23 -prm 3 -out 1 –ini 200 –res PFHG ex5.pdb

The original segment of 7 residues is now replaced by a new segment of residue “PFHG”, and the conformation of the new segment is then predicted. This is a deletion example.

The following command is an insertion case:

$ loopy –obj 17-23 -prm 3 -out 1 –ini 200 –res PFHGAAAAA ex5.pdb

Log File

When you run loopy program with the command:

$loopy –obj 17-23 -ini 100 –res AHGGF ex1.pdb

the log file shown on the screen will display information during program execution .

The log file is as follows with comments bold in bracket:

```
reading from file: ../library/back_small_rotamer

from back_small_rotamer: the number copies for residue: A 4
from back_small_rotamer: the number copies for residue: C 7
from back_small_rotamer: the number copies for residue: D 7
from back_small_rotamer: the number copies for residue: E 6
from back_small_rotamer: the number copies for residue: F 5
from back_small_rotamer:the number copies for residue: G 11
from back_small_rotamer: the number copies for residue: H 8
from back_small_rotamer: the number copies for residue: I 4
from back_small_rotamer: the number copies for residue: K 5
from back_small_rotamer: the number copies for residue: L 5
from back_small_rotamer: the number copies for residue: M 5
from back_small_rotamer: the number copies for residue: N 8
from back_small_rotamer: the number copies for residue: P 1
from back_small_rotamer: the number copies for residue: Q 6
from back_small_rotamer: the number copies for residue: R 6
from back_small_rotamer: the number copies for residue: S 7
from back_small_rotamer: the number copies for residue: T 5
from back_small_rotamer: the number copies for residue: V 5
from back_small_rotamer: the number copies for residue: W 4
from back_small_rotamer: the number copies for residue: Y 6

reading from file: ../library/side_small_rotamer

from side_small_rotamer: the number copies for residue: A 1
from side_small_rotamer: the number copies for residue: C 3
from side_small_rotamer: the number copies for residue: D 5
from side_small_rotamer:the number copies for residue: E 12
from side_small_rotamer: the number copies for residue: F 4
from side_small_rotamer: the number copies for residue: G 1
from side_small_rotamer: the number copies for residue: H 9
from side_small_rotamer: the number copies for residue: I 7
from side_small_rotamer:the number copies for residue: K 49
from side_small_rotamer: the number copies for residue: L 9
from side_small_rotamer:the number copies for residue: M 17
from side_small_rotamer:the number copies for residue: N 11
from side_small_rotamer: the number copies for residue: P 3
from side_small_rotamer:the number copies for residue: Q 19
```

```
from side_small_rotamer:the number copies for residue: R 39
from side_small_rotamer: the number copies for residue: S 3
from side_small_rotamer: the number copies for residue: T 3
from side_small_rotamer: the number copies for residue: V 3
from side_small_rotamer: the number copies for residue: W 8
from side_small_rotamer: the number copies for residue: Y 8

reading pdb file:ex1.pdb

loopy is going to use the chain with empty space ' ' as the
default to define the loop
number of residues in loop from 17 to 23 does not match the
sequence given
there are residue deletion...
replace the original segment with a new segment: AHGGF
generating loop candidate conformation...
number of segments generated :333
number of segment successfully connected: 100
minimizing loop candidates...
discarding bad loop candidates...
add side-chains to candidate conformations...
minimizing again...
discarding bad candidates...
loop fusion to generate more candidates...
loop fusion to generate more candidates...
do side-chain prediction again...
minimizing again...
loop fusion to generate more candidates...
loop fusion to generate more candidates...
discarding bad candidates...
output final result...ex1_loopy.pdb
```

5.4) profix

Introduction
Commands
Tutorial
Log File

Introduction

a) What is Profix?

Profix is a program to reconstruct protein missing side-chain atoms, protein missing backbone atoms and protein missing residues. The profix program uses backbone rotamer library, side-chain rotamer library and distance geometry constraints to sample segment conformations to recover missing atoms either on side-chain or on backbone.

For every residue of a protein, profix compares the atoms with the standard residue and find out if there are any atom missing. If the missing atoms are side-chain atoms, profix tries to sample side-chain rotamer library and identify the side-chain rotamer that has the closest conformation to the original side-chain; if there are no atoms left on the original side-chains, profix chooses the side-chain rotamer of the lowest energy. If the missing atoms are backbone atoms, profix tries to sample backbone rotamer library and identify one that can best connect the two neighboring residues.

Missing residues (residues existing in SEQRES card while missing in ATOM card) are reconstructed by built-in nest and scap module.

In recovering missing atoms, atoms may deviate a little bit from their original positions. The deviation is usually less than 0.2 angstrom. It happens because the new rotamer, which is used to replace the incomplete residue, may not have the same bond angle and length as in the original residue.

Commands

For all the programs included in JACKAL package, you can find their usage by running the program with no arguments.

After unpacking the jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/profix. This directory contains all examples used in this tutorial for running profix program. Go to this directory and type:

$profix

profix will print out message about its usage as following:

```
**************************************************************
profix is to fix missing atoms and residues in pdb file
the missing residues decided by comparing SEQRES card and ATOM card
comments? send to Jason Xiang at jsxzx@yahoo.com
**************************************************************

USAGE: pdbfix -prm num -fix num file.pdb
-prm select  atom model.default is 1.
-prm 1,all atom model
-prm 2,heavy atom model
-fix what to be fixed? default is 1.
-fix  0, fix missing atoms including missing atoms in backbone and sidechain
-fix  1, besides option 0, also fix missing residues by reading PDB SEQRES card
```

When you run profix, you type: profix plus option plus the integer value for the option plus the pdb file. If there is something wrong with your command, profix will exit and print out an error message. The above options can be omitted, where profix uses their default values; or several options can be used at the same time.

prm option
The option "-prm num" is used to decide which atom model is used, either all atom model or heavy atom model. In the case of all-atom model, missing protons will first be reconstructed onto the protein; prediction is then made to those residues of missing heavy atoms. In the heavy atom model, protons are stripped from all the residues in the protein.

fix option
fix option is used to tell profix what to be fixed. The option 0 is used to recover missing atoms either on side-chain or backbone, i.e., at least one atom must exist in the original residue otherwise the residue is missing. Option 1 is used not only to recover missing atoms but also to recover missing residues. Missing residues are defined by those residues existing in SEQRES card while not existing in ATOM card. The missing residues are inserted back using nest module.

Tutorial

The tutorial of profix can be found in the directory jackal/profix

1) **Example 1: reconstruction of missing atoms on side-chains**

The first example is the case that some atoms are missing in a number of residues. Ex1.pdb has 46 residues and some atoms on side-chains are missing on the first 4 residues as following:

```
ATOM      1  N    THR     1      16.864  14.059   3.442
ATOM      2  CA   THR     1      16.868  12.814   4.233
ATOM      3  C    THR     1      15.583  12.775   4.990
ATOM      4  O    THR     1      15.112  13.824   5.431
ATOM      5  CB   THR     1      18.060  12.807   5.200
ATOM      7  CG2  THR     1      18.107  11.671   6.155
ATOM      8  N    THR     2      15.050  11.574   5.153
ATOM      9  CA   THR     2      13.832  11.480   5.936
ATOM     10  C    THR     2      14.140  10.771   7.259
ATOM     11  O    THR     2      14.981   9.844   7.352
ATOM     12  CB   THR     2      12.704  10.758   5.178
ATOM     15  N    CYS     3      13.415  11.193   8.281
ATOM     16  CA   CYS     3      13.564  10.664   9.653
ATOM     17  C    CYS     3      12.179  10.373  10.192
ATOM     18  O    CYS     3      11.303  11.249   9.999
ATOM     21  N    CYS     4      11.998   9.281  10.858
ATOM     22  CA   CYS     4      10.657   8.927  11.341
ATOM     23  C    CYS     4      10.648   8.749  12.842
ATOM     24  O    CYS     4      11.615   8.257  13.440
ATOM     25  CB   CYS     4      10.137   7.678  10.634
```

Where OG1 is missing in T1, OG1 and CG2 missing in T2, CB and SG missing in C3, and SG missing in C4.

With this command:

$ profix –prm 2 -fix 0 ex1.pdb

The above command line means recovering all missing atoms with heavy-atom model. The final output is ex1_fix.pdb, which has all missing atoms reconstructed.

Or if with this command:
$ profix –prm 1 -fix 0 ex1.pdb

The output ex1_fix.pdb will not only recover all missing atoms but also missing protons.

There is no limit on the number of residues where atoms are missing.

2) Example 2: reconstruction of missing atoms on backbone

The second example is the case that some backbone atoms are missing in a number of residues. Ex2.pdb also has 46 residues and some atoms on backbone are missing on several residues as following:

```
ATOM    216  CA  ARG A  31      15.065  -3.131 -11.564
ATOM    219  CB  ARG A  31      15.808  -2.328 -10.487
ATOM    220  CG  ARG A  31      17.340  -2.395 -10.667
ATOM    221  CD  ARG A  31      18.019  -1.574  -9.618
ATOM    222  NE  ARG A  31      19.477  -1.649  -9.716
ATOM    223  CZ  ARG A  31      20.242  -2.599  -9.191
ATOM    224  NH1 ARG A  31      19.728  -3.536  -8.391
ATOM    225  NH2 ARG A  31      21.549  -2.616  -9.473
ATOM    226  N   MET A  32      12.813  -2.225 -11.590
ATOM    230  CB  MET A  32      10.794  -0.812 -11.538
ATOM    231  CG  MET A  32       9.331  -0.839 -11.126
ATOM    232  SD  MET A  32       8.582   0.705 -11.695
ATOM    233  CE  MET A  32       8.605   0.510 -13.472
ATOM    238  CB  PHE A  33      10.618  -4.268 -15.660
ATOM    239  CG  PHE A  33      10.060  -2.993 -16.221
ATOM    240  CD1 PHE A  33       8.800  -2.541 -15.824
ATOM    241  CD2 PHE A  33      10.812  -2.254 -17.131
ATOM    242  CE1 PHE A  33       8.271  -1.372 -16.327
ATOM    243  CE2 PHE A  33      10.287  -1.055 -17.655
ATOM    244  CZ  PHE A  33       9.020  -0.625 -17.247
```

In ex2.pdb as shown above, N and O are missing from R31, CA and O missing from M32, N, CA, O missing from F33.

With the following command:

```
$ profix –prm 2  -fix 0  ex2.pdb
```

The output is ex2_fix.pdb and all missing atoms have been reconstructed.

There is no limit on the number of problematic residues.

3) Example 3: reconstruction of missing atoms anywhere in a number of residues

Ex3.pdb is the case that has atoms missing on side-chains or backbones throughout the pdb. No matter where the atoms are missing, profix will try to reconstruct the missing atoms by the simple command:

$ profix –prm 2 -fix 0 ex3.pdb

4) Example 4: reconstruction of missing residues

Ex4.pdb has two residues without coordinates. The two residues V8 and A9 appear in the SEQRES card but do not show up in ATOM card. Besides the two residues missing, there are also missing atoms on side-chain and backbone in residue S11 and N12.

However, profix will fix all these problems with the command:

$ profix –prm 2 -fix 1 ex4.pdb

the option "–fix 1" is used to reconstruct missing residues in addition to "-fix 0".

5) Example 5: reconstruction of complete pdb from CA only

Example 5 is the case where only CA atoms exist in the original structure.
Below is ex5.pdb file:

```
ATOM      2  CA  THR     1      16.868  12.814   4.233
ATOM      9  CA  THR     2      13.832  11.480   5.936
ATOM     16  CA  CYS     3      13.564  10.664   9.653
ATOM     22  CA  CYS     4      10.657   8.927  11.341
ATOM     28  CA  PRO     5       9.474   9.025  14.942
ATOM     35  CA  SER     6       8.706   5.341  15.230
ATOM     41  CA  ILE     7       8.941   2.087  13.267
ATOM     49  CA  VAL     8       5.250   2.180  12.300
ATOM     56  CA  ALA     9       5.600   5.711  11.016
ATOM     61  CA  ARG    10       8.470   4.577   8.764
ATOM     72  CA  SER    11       6.494   1.547   7.538
ATOM     78  CA  ASN    12       3.543   3.842   6.717
ATOM     86  CA  PHE    13       5.885   6.300   5.014
ATOM     97  CA  ASN    14       7.273   3.580   2.775
ATOM    105  CA  VAL    15       3.781   2.488   1.703
ATOM    112  CA  CYS    16       2.845   6.154   1.083
ATOM    118  CA  ARG    17       5.821   6.448  -1.247
ATOM    129  CA  LEU    18       4.945   3.388  -3.380
```

With the command:

Profix –prm 2 –fix 0 ex5.pdb

The output ex5_fix.pdb will add all missing atoms.

Log File

When you run the profix program with the command:

$profix –prm 2 –fix 1 ex4.pdb

```
reading from file: ../library/back_large_rotamer
from back_large_rotamer: the number copies for residue:A 65
from back_large_rotamer: the number copies for residue:C 73
from back_large_rotamer: the number copies for residue:D 91
from back_large_rotamer: the number copies for residue:E 69
from back_large_rotamer: the number copies for residue:F 69
from back_large_rotamer:the number copies for residue:G 153
from back_large_rotamer: the number copies for residue:H 63
from back_large_rotamer: the number copies for residue:I 52
from back_large_rotamer: the number copies for residue:K 70
from back_large_rotamer: the number copies for residue:L 63
from back_large_rotamer: the number copies for residue:M 57
from back_large_rotamer:the number copies for residue:N 104
from back_large_rotamer:the number copies for residue: P 25
from back_large_rotamer:the number copies for residue: Q 65
from back_large_rotamer:the number copies for residue: R 64
from back_large_rotamer:the number copies for residue: S 88
from back_large_rotamer:the number copies for residue: T 70
from back_large_rotamer:the number copies for residue: V 51
from back_large_rotamer:the number copies for residue: W 51
from back_large_rotamer:the number copies for residue: Y 70

reading from file: ../library/side_large_rotamer
from side_large_rotamer: the number copies for residue: A 1
from side_large_rotamer: the number copies for residue: C 9
from side_large_rotamer:the number copies for residue:D 112
from side_large_rotamer:the number copies for residue:E 821
from side_large_rotamer: the number copies for residue:F 55
from side_large_rotamer: the number copies for residue: G 1
from side_large_rotamer:the number copies for residue:H 125
from side_large_rotamer:the number copies for residue: I 71
from side_large_rotamer:the numbercopies for residue:K 1948
from side_large_rotamer: the number copies for residue:L 69
from side_large_rotamer:the number copies for residue:M 391
from side_large_rotamer:the number copies for residue:N 175
from side_large_rotamer: the number copies for residue: P 9
from side_large_rotamer:the number copies for residue:Q 785
from side_large_rotamer:thenumber copies for residue:R 1730
from side_large_rotamer:the number copies for residue: S 11
from side_large_rotamer: the number copies for residue: T 9
from side_large_rotamer: the number copies for residue: V 7
from side_large_rotamer:the number copies for residue: W 93
from side_large_rotamer:the number copies for residue: Y 65

reading from file: ../library/side_small_rotamer
from side_small_rotamer: the number copies for residue: A 1
```

```
from side_small_rotamer: the number copies for residue: C 3
from side_small_rotamer: the number copies for residue: D 5
from side_small_rotamer:the number copies for residue: E 12
from side_small_rotamer: the number copies for residue: F 4
from side_small_rotamer: the number copies for residue: G 1
from side_small_rotamer: the number copies for residue: H 9
from side_small_rotamer:the number copies for residue: I 7
from side_small_rotamer:the number copies for residue: K 49
from side_small_rotamer:the number copies for residue: L 9
from side_small_rotamer:the number copies for residue: M 17
from side_small_rotamer:the number copies for residue: N 11
from side_small_rotamer: the number copies for residue: P 3
from side_small_rotamer:the number copies for residue: Q 19
from side_small_rotamer:the number copies for residue: R 39
from side_small_rotamer: the number copies for residue: S 3
from side_small_rotamer: the number copies for residue: T 3
from side_small_rotamer: the number copies for residue: V 3
from side_small_rotamer: the number copies for residue: W 8
from side_small_rotamer: the number copies for residue: Y 8

reading file:../library/bound_nearnext1168.dist
reading file:../library/bound_hbondind1168.dist
reading pdb file:ex4.pdb

checking missing atoms...
Atom missing: S11-- N  !
Atom missing: S11-- C  !
Atom missing: S11-- O  !
Atom missing: N12-- O  !
Atom missing: N12-- CB !
Atom missing: N12-- CG !
Atom missing: N12-- OD1!
Atom missing: N12-- ND2!
the total number of missing atoms: 8
add back missing atoms...
not standard residue in hook! S10
not standard residue in hook! S10
[hook means there are backbone atoms missing in neighboring
residues, the two atoms could not be naturally connected]

[below is scap out]
calculate solvent accessible surface....
building multiple rotamers for each side-chain...
building random initial conformation...

[below is the output from nest module]
check error in alignment blocks...



Warning......
the pdb file:ex4 has breaker at:F13 N14
```

```
checking missing atoms...
the total number of missing atoms: 0
there is no missing atoms.

find corresponding residues between pdb and pir for the
structure pir:unknown

**********warning!**********
error in specifying the start and end residue ids in the
pir alignment:
>P1;unknown
structure:unknown: :0: :0
TTCCPSIVARSNFNVCRLPGTPEALCATYTGCIIIPGATCPGDYAN
the program fixs it anyway by doing alignments between
residues in pdb and in pir....

performing Needleman-Wunsch alignment to find the
corresponding residue
in pdb file and in pir file...

 TTCCPSI--R SNFXXNVCRL PGTPEALCAT YTGCIIIPGA TCPGDYAN
 TTCCPSIVAR SNF--NVCRL PGTPEALCAT YTGCIIIPGA TCPGDYAN
 *******  * ***  ***** ********** ********** ********

warning! residue: V at position: 7 does not exist in PDB
deleting this residue from the alignment...

warning! residue: A at position: 8 does not exist in PDB
deleting this residue from the alignment...

detecting secondary structure regions using dssp
definition...

T0 -
T1 e
C2 e
C3 -
P4 -
S5 -
I6 -
R7 -
S8 h
N9 h
F10 h
N13 h
V14 h
C15 h
R16 h
L17 -
P18 -
G19 -
T20 -
P21 -
E22 h
```

```
A23 h
L24 h
C25 h
A26 h
T27 h
Y28 h
T29 h
G30 -
C31 -
I32 e
I33 e
I34 -
P35 -
G36 -
A37 -
T38 -
C39 -
P40 -
G41 -
D42 -
Y43 -
A44 -
N45 -

check error in composite pirs...

find out chain breakers and resolve...

calculate sequence conserve score at each residue between
sequence and structure pirs...
conserve score  0 T---T:0.614368
conserve score  1 T---T:0.750376
conserve score  2 C---C:0.850721
conserve score  3 C---C:0.897388
conserve score  4 P---P:0.933912
conserve score  5 S---S:0.885297
conserve score  6 I---I:0.819727
conserve score  7 R---V:0.675613
conserve score  8 S---A:0.578764
conserve score  9 N---R:0.535237
conserve score  10 F---S:0.50321
conserve score  11 ----N:0.452548
conserve score  12 ----F:0.518995
conserve score  13 N---N:0.689008
conserve score  14 V---V:0.78458
conserve score  15 C---C:0.850721
conserve score  16 R---R:0.931592
conserve score  17 L---L:1
conserve score  18 P---P:1
conserve score  19 G---G:1
conserve score  20 T---T:1
conserve score  21 P---P:1
conserve score  22 E---E:1
conserve score  23 A---A:1
```

```
conserve score  24 L---L:1
conserve score  25 C---C:1
conserve score  26 A---A:1
conserve score  27 T---T:1
conserve score  28 Y---Y:1
conserve score  29 T---T:1
conserve score  30 G---G:1
conserve score  31 C---C:1
conserve score  32 I---I:1
conserve score  33 I---I:1
conserve score  34 I---I:1
conserve score  35 P---P:1
conserve score  36 G---G:1
conserve score  37 A---A:1
conserve score  38 T---T:1
conserve score  39 C---C:1
conserve score  40 P---P:1
conserve score  41 G---G:1
conserve score  42 D---D:0.931592
conserve score  43 Y---Y:0.850721
conserve score  44 A---A:0.750376
conserve score  45 N---N:0.614368

model building for the 0th alignment blocks


finish initial model building...
write down the final structure...ex4_fix.pdb
```

5.5) ctrip

Introduction
Commands
Tutorial
Log File

Introduction

a) What is Ctrip?

Ctrip is a program to reconstruct protein complete structure from CA trace atoms. The ctrip program uses backbone rotamer library, side-chain rotamer library and distance geometry constraints to sample segment conformations created on fly. The best segment conformation is that which is most likely to observe the CA trace.

In ctrip, the large backbone and side-chain rotamer libraries are used. You can find out the two libraries in the directory jackal/library with the name: back_large_rotamer and side_large_rotamer. Back_large_rotamer has 1413 backbone rotamers and side_large_rotamer has more than 6000 side-chain rotamers. The large backbone rotamer library comes from the 135 proteins with 10 degree cutoff and the large side-chain rotamer comes from 297 proteins with 10 degree cutoff. They represent 96% of all rotamers in 135 and 297 chains respectively.

Given a protein only with CA trace atoms, ctrip is first to pick two random backbone rotamers for the last two residues. The two rotamers are linked and superimposed onto the two CA atoms of the original protein. The best rotamers are those which could best link the two CA atoms. The procedure moves on to the residue next to the last. After the iteration is finished at the first residue of the protein, ctrip then calculates the hydrogen bonds among backbone atoms for every residue in the protein. The definition of hydrogen bond is based on the statistical distance between the atoms N and C, and between CA and C. If the two distances are both within the cutoff of the upper and lower bounds of a regular hydrogen bond, a hydrogen bond is considered existing between the two residues. With the hydrogen bonds identified among all residues, the secondary structure of the protein is defined.

With the knowledge of the secondary structure of the protein, the backbone atoms of the protein are then predicted again with the same procedure, however, with constraints on backbone rotamers that are possible to satisfy the secondary structure definition. The final structure is then subjected to scap program to have side-chains assembled.

Commands

For all the programs included in JACKAL package, you can find their usage by running the program with no arguments.

After unpacking the jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/ctrip. This directory contains all examples used in this tutorial for running ctrip program. Go to this directory and type:

$ctrip

ctrip will print out message about its usage as following:

```
**************************************************
ctrip is to build complete structure based on CA only
comments? send to Jason Xiang at jsxzx@yahoo.com
**************************************************

USAGE: ctrip -prm num -k num -fast num file.pdb
-prm   select  atom model.default is 1
-prm  1,all atom model
-prm  2,heavy atom model
-prm  3,backbone atom model
-k   is original atoms kept beyond CA? default is 1.
-k   0, delete all original atoms except CA
-k   1, keep all original atoms
```

When you run ctrip, you type: ctrip plus option plus the integer value for the option plus the pdb file. If there is something wrong with your command, ctrip will exit and print out an error message. The above options can be omitted, where ctrip uses their default values; or several options can be used at the same time.

prm option
Option "-prm num" is used to decide which atom model is used, either all atom model or heavy atom model or backbone atom model. In the case of all-atom model, the complete structure constructed from the rotamer library is of all protons already assembled. In the case of heavy atom model, protons are stripped off from all rotamers first, and the complete structure constructed in this way is thus of no protons. In the backbone model, no side-chain rotamer library is used and the side-chains are stripped off from the backbone rotamer. No protons will be included in this model.

k option
The option is used to tell ctrip if the original atoms other than CA should be stripped off or keep in intact. With option 1, all original atoms are kept; otherwise, all original atoms

are stripped off except CA atom. If all the original atoms are kept, ctrip program will only work on those residues with missing backbone or side-chain atoms.

Tutorial

The tutorial of ctrip can be found in the directory jackal/ctrip

4) **Example 1: reconstruction of a complete structure from CA only**

The first example is the case that only CA atoms exist in ex1.pdb.

With this command:

$ ctrip –prm 2 -k 0 ex1.pdb

The above command line means deleting all the original atoms except CA and reconstructing the complete structure based on CA atoms only. Since ex1.pdb happens to be CA atom only, so deletion is ignored. The output is a complete structure with all heavy atoms called ex1_fix.pdb

Or if with this command:

$ ctrip –prm 1 -k 0 ex1.pdb

The output ex1_fix.pdb is a complete structure will all protons assembled.

5) Example 2: reconstruction of missing atoms on backbone

The second example is the case that some backbone atoms are missing in a number of residues. Ex2.pdb also has 46 residues and some atoms on backbone are missing on several residues as following:

```
ATOM    216  CA  ARG A  31      15.065  -3.131 -11.564
ATOM    219  CB  ARG A  31      15.808  -2.328 -10.487
ATOM    220  CG  ARG A  31      17.340  -2.395 -10.667
ATOM    221  CD  ARG A  31      18.019  -1.574  -9.618
ATOM    222  NE  ARG A  31      19.477  -1.649  -9.716
ATOM    223  CZ  ARG A  31      20.242  -2.599  -9.191
ATOM    224  NH1 ARG A  31      19.728  -3.536  -8.391
ATOM    225  NH2 ARG A  31      21.549  -2.616  -9.473
ATOM    226  N   MET A  32      12.813  -2.225 -11.590
ATOM    230  CB  MET A  32      10.794  -0.812 -11.538
ATOM    231  CG  MET A  32       9.331  -0.839 -11.126
ATOM    232  SD  MET A  32       8.582   0.705 -11.695
ATOM    233  CE  MET A  32       8.605   0.510 -13.472
ATOM    238  CB  PHE A  33      10.618  -4.268 -15.660
ATOM    239  CG  PHE A  33      10.060  -2.993 -16.221
ATOM    240  CD1 PHE A  33       8.800  -2.541 -15.824
ATOM    241  CD2 PHE A  33      10.812  -2.254 -17.131
ATOM    242  CE1 PHE A  33       8.271  -1.372 -16.327
ATOM    243  CE2 PHE A  33      10.287  -1.055 -17.655
```

```
ATOM    244  CZ  PHE A  33       9.020  -0.625 -17.247
```

In ex2.pdb as shown above, N and O are missing from R31, CA and O missing from M32, N, CA, O missing from F33.

With the following command:

$ ctrip –prm 2 -fix 0 ex2.pdb

The output is ex2_fix.pdb and all missing atoms have been reconstructed.

There is no limit on the number of problematic residues.

6) Example 3: reconstruction of missing atoms anywhere in a number of residues

Ex2.pdb is the case that has some residues missing backbone atoms and some residues complete with all backbone atoms. In ex2.pdb, T2 and C3 have only CA atoms while all other residues are fine.

With the command:

$ ctrip –prm 1 -k 1 ex2.pdb

Ctrip will keep all the original atoms and try to work only on T2 and C3 residues. The output ex2_fix.pdb is a complete structure with all protons.

However, with this command:

$ ctrip –prm 1 -k 0 ex2.pdb

All atoms in ex2.pdb will be deleted except CA atoms. The output is a new structure based on CA trace only.

Log File

When you run the ctrip program with the command:

$ctrip –prm 1 –k 1 ex2.pdb

```
reading from file: ../library/back_large_rotamer

from back_large_rotamer: the number copies for residue:A 65
from back_large_rotamer: the number copies for residue:C 73
from back_large_rotamer: the number copies for residue:D 91
from back_large_rotamer: the number copies for residue:E 69
from back_large_rotamer: the number copies for residue:F 69
from back_large_rotamer:the number copies for residue:G 153
from back_large_rotamer: the number copies for residue:H 63
from back_large_rotamer: the number copies for residue:I 52
from back_large_rotamer: the number copies for residue:K 70
from back_large_rotamer: the number copies for residue:L 63
from back_large_rotamer: the number copies for residue:M 57
from back_large_rotamer:the number copies for residue:N 104
from back_large_rotamer:the number copies for residue: P 25
from back_large_rotamer:the number copies for residue: Q 65
from back_large_rotamer:the number copies for residue: R 64
from back_large_rotamer:the number copies for residue: S 88
from back_large_rotamer:the number copies for residue: T 70
from back_large_rotamer:the number copies for residue: V 51
from back_large_rotamer:the number copies for residue: W 51
from back_large_rotamer:the number copies for residue: Y 70

reading from file: ../library/side_large_rotamer

from side_large_rotamer: the number copies for residue: A 1
from side_large_rotamer: the number copies for residue: C 9
from side_large_rotamer:the number copies for residue:D 112
from side_large_rotamer:the number copies for residue:E 821
from side_large_rotamer: the number copies for residue:F 55
from side_large_rotamer: the number copies for residue: G 1
from side_large_rotamer:the number copies for residue:H 125
from side_large_rotamer:the number copies for residue: I 71
from side_large_rotamer:the numbercopies for residue:K 1948
from side_large_rotamer: the number copies for residue:L 69
from side_large_rotamer:the number copies for residue:M 391
from side_large_rotamer:the number copies for residue:N 175
from side_large_rotamer: the number copies for residue: P 9
from side_large_rotamer:the number copies for residue:Q 785
from side_large_rotamer:thenumber copies for residue:R 1730
from side_large_rotamer:the number copies for residue: S 11
from side_large_rotamer: the number copies for residue: T 9
from side_large_rotamer: the number copies for residue: V 7
from side_large_rotamer:the number copies for residue: W 93
```

```
from side_large_rotamer:the number copies for residue: Y 65

reading from file: ../library/side_small_rotamer

from side_small_rotamer: the number copies for residue: A 1
from side_small_rotamer: the number copies for residue: C 3
from side_small_rotamer: the number copies for residue: D 5
from side_small_rotamer:the number copies for residue: E 12
from side_small_rotamer: the number copies for residue: F 4
from side_small_rotamer: the number copies for residue: G 1
from side_small_rotamer: the number copies for residue: H 9
from side_small_rotamer:the number copies for residue: I 7
from side_small_rotamer:the number copies for residue: K 49
from side_small_rotamer:the number copies for residue: L 9
from side_small_rotamer:the number copies for residue: M 17
from side_small_rotamer:the number copies for residue: N 11
from side_small_rotamer: the number copies for residue: P 3
from side_small_rotamer:the number copies for residue: Q 19
from side_small_rotamer:the number copies for residue: R 39
from side_small_rotamer: the number copies for residue: S 3
from side_small_rotamer: the number copies for residue: T 3
from side_small_rotamer: the number copies for residue: V 3
from side_small_rotamer: the number copies for residue: W 8
from side_small_rotamer: the number copies for residue: Y 8

reading file:../library/bound_nearnext1168.dist


reading file:../library/bound_hbondind1168.dist


reading pdb file:ex2.pdb

checking missing atoms...
Atom missing: T2-- N  !
Atom missing: T2-- C  !
Atom missing: T2-- O  !
Atom missing: T2-- CB !
Atom missing: T2-- OG1!
Atom missing: T2-- CG2!
Atom missing: T2-- HN !
Atom missing: T2-- HA !
Atom missing: T2-- HB !
Atom missing: T2--1HG2!
Atom missing: T2--2HG2!
Atom missing: T2--3HG2!
Atom missing: C3-- N  !
Atom missing: C3-- C  !
Atom missing: C3-- O  !
Atom missing: C3-- CB !
Atom missing: C3-- SG !
Atom missing: C3-- HN !
Atom missing: C3-- HA !
Atom missing: C3--1HB !
```

```
Atom missing: C3--2HB !
the total number of missing atoms: 21
add back missing atoms...
calculate solvent accessible surface....
building multiple rotamers for each side-chain...
building random initial conformation...
writing output file:ex2_fix.pdb
```

5.6) addh

Introduction
Commands
Tutorial

Introduction

a) What is addh?

Addh is a program to reconstruct protons given a protein structure of heavy atoms. The protons are subjected to local energy minimization besides standard geometry constraints.

Commands

For all the programs included in the JACKAL package, you can find their usage by running the program with no arguments.

After unpacking jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/addh. This directory contains all examples used in this tutorial for running addh program. Go to this directory and type:

$addh

addh will print out message about its usage as following:

```
********************************************************** **
addh is a program of adding protons
questions? please refer to Dr.Jason Z. Xiang at zx11@columbia.edu
********************************************************** ***
Usage:
addh -k file.pdb
-k: keep original hydrogen atoms.
-k 1:keep original protons.default is 1
-k 0:delete original protons
```

When you run addh, you type: addh plus option plus the integer value for the option plus the pdb file. If there is something wrong with your command, addh will exit and print out an error message. The above options can be omitted, where addh uses their default values; or several options can be used at the same time.

k option

The option "-k num" is used to decide if the original protons should be stripped or kept. With the option 1, the original protons will be kept intact; with option 0, the original protons will be discarded and new protons will be predicted.

Tutorial

The tutorial of addh can be found in the directory jackal/addh

1) **Example 1: reconstruction of all protons**

With this command:

$ addh -k 0 ex1.pdb

The above command line means first stripping all protons from ex1.pdb and then assembling new protons.

Or if with this command:

$ addh –k 1 -ex1.pdb

which means keeping the original protons and predicting missing protons.

However, if the residue is not complete, the protons may not be successfully added. In that case, an error message will be printed out. For example, if the protein is of CA only, no protons can be added since there is no complete backbone to hold any protons.

5.7) conref

Introduction
Commands
Tutorial

Introduction

a) What is ConRef?

ConRef is a constrained structure refinement program. The constraints include hydrogen network and backbone framework of the original structure, i.e., there is an energy penalty when a new sampled structure breaks an existing hydrogen bond or has a large rmsd from the original. This is to make sure that conformational sampling is done within the conformational space similar to the original one. The constraints can be relaxed with more rounds of refinement.

The structure refinement program can refine in three levels: removing clashing atoms, refinement in all loop regions and refinement in all regular secondary regions plus loop regions. Energy minimization by removing atom clashes is performed using our fast torsion angle minimizer with smoothing energy techniques. Refinement in loop regions is done using our loopy program and refinement in side-chain conformation is performed by our side-chain program scap. Refinement in helix or sheet regions is done by a procedure similar to loopy while the hydrogen constraints in the regular secondary regions are applied so that the refinement will not disrupt the original hydrogen network. The hydrogen bond has the same definition as in the DSSP program. In the refinement, the conformational sampling is restricted with a rmsd to the original structure.

Commands

For all the programs included in the JACKAL package, you can find their usage by running the program with no arguments.

After unpacking the jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/conref. This directory contains all examples used in this tutorial for running conref program. Go to this directory and type:

$conref

conref will print out the message about its usage as follows:

```
**************************************************************************
conref is a program to refine protein structures with constraints from the original
structure

questions? please refer to Dr.Jason Z. Xiang at zx11@columbia.edu.
**************************************************************************
Usage: conref -prm num -seed num -fast num  -opt num -nopt num -obj num-num -rmsd
float -out num -cid char file.pdb
-fast     fast mode.from 0-5; default is 3.with 5 fastest
-opt      refine mode, from 1-3; default is 3.with 3 most thorough refinement
-opt   1, remove atom clashes;
-opt   2, refine in all loop regions;
-opt   3, refine in all loop and secondary regions
-seed     set seed number, defautl is 18120.
-prm      force field parameter mode. from 1-4, default is 3.
-prm    1, charmm all atoms;
-prm    2, amber all atoms;
-prm    3, charmm heavy atoms;
-prm   4, amber heavy atoms
-nopt     number of rounds of refinement.default is 1.
-out      output mode, from 0-2, defautl is 0. with 2 all intermediate output
-out   0, only final structures written down;
-out   1, intermediate structures also outputted;
-out   2, more intermediates output
-cid       chain id.
-obj       specify the segment to be minimized; default is whole chain.
-rmsd     rmsd restraint.default is 2.0A
file.pdb  pdb file
```

When you run conref, you type: conref plus option plus the integer value for the option plus the pdb file. If there is something wrong with your command, conref will exit and print out an error message. The above options can be omitted and conref uses their default values; or several options can be used at the same time.

fast option
The option "-fast num" is used to decide how fast the conref program should run. The fast mode has 6 levels from 0 to 5 with 5 as the fastest mode. The default is 3.

seed option
Seed option is just to give a random seed number to the conref program. The default is 18120.

out option
The out option is used to direct conref about how many structures should be outputted, such as, if the intermediate structures should be written down to hard disk.

When "-out 0" is used, the conref program will only write down the final model; when "-out 1" is used, the conref program will also write down the intermediate structures; when "-out 2" is used, the conref program will output all the intermediate structures in the process of refinement.

opt option
The opt option is used to optimize and refine the original structure. There are 3 levels of refinement with 3 as the most thorough refinement. The default level is 3. The option 1 is to remove atom clashes. The option 2 is to refine the loop regions. When the refinement is applied to a specific region, conref will first sample a large number of conformations around the original conformation and discard those conformations with rmsd larger than a specific value (2.0A is default) away from the original conformation. So the refinement is always performed with constraints around the original structure framework. However, such kind of constraints can be circumvented by several rounds of refinement as explained below. In the process of refinement in the helix and sheet regions, the hydrogen bonding network from the original structure is applied so that the final structure after refinement will also observe the original hydrogen network. However, in the refinement of the loop regions, the original loop regions may refine into helix or sheet depending on the energy criteria, though this does not usually happen.

Each sampled conformation will be subjected to our fast torsion space minimizer with smoothing energy technique.

nopt option
The nopt option is used to decide how many rounds of refinement are required. In the first round of refinement, the refinement feels pressure of framework constraints from the original structures; in the second round of refinement, the refinement is performed based on the output from the first round, so that the framework constraint pressure is different than that in the original structure. Though in each round of refinement, the sampled

conformation is restricted to within a specfic rmsd value (2.0A as default) from their starting point. This restriction will be more and more relaxed compared with the original structure with more rounds of refinement. However, it does not necessarily mean more rounds of refinement, usually further away from the original starting point, will bring the structure closer to the native, though energetically it is lower.

prm option
The option "-prm num" is used to decide which force field is used, either CHARMM or AMBER and which atom model is used, either all-atom model or heavy-atom model. With option 1 or 3, the CHARMM22 force field will be used, i.e., torsion energy, van der Waals radius and charge parameters taken from CHARMM. In the case of all atom model, the missing protons will first be added back to each residue in the protein wherever they should exist and then loopy is executed on the protein with all hydrogen atoms assembled. In the case of the heavy atom model, all protons on the protein will first be stripped and then loopy is executed. In the heavy-atom model, the charge of hydrogen atoms will give back to their parent heavy atoms so that functional group still keep the same amount of total charges.

cid option
cid option is used to specify the chain to work on. The default is the chain with id empty.

rmsd option
rmsd option is used to specify the rmsd cutoff in conformations sampling. The default is 2.0A, which means only those conformations within 2.0A rmsd from the original structure are retained in the process of conformations sampling.

obj option
This option is used to specify the region to work on. The default is the whole chain.

Tutorial

The tutorial of conref can be found in the directory jackal/conref

1) **Example 1: refinement of all loop regions**

The first example is the case that uses conref refine all loop regions.

With this command:

$ conref – opt 2 ex1.pdb

The above command line means refinement in all loop regions in the chain with empty id of ex1.pdb with rmsd cutoff 2.0 and one round of refinement is required. The atom-model is all heavy atom. Only the final structure is to be outputted.

Or if with this command:

$ conref – opt 2 -nopt 2 –rmsd 3.0 -cid B ex1.pdb

it means refinement is for chain B with rmsd cutoff 3.0A and two rounds of refinement is required.

The final output is ex1_final.pdb

2) Example 2: refinement of all regions.

The second example uses conref to refine all regions in ex2.pdb.

With this command:

$ conref – opt 3 ex2.pdb

The above command line means refinement in all loop regions in the chain of empty id of ex1.pdb with rmsd cutoff 2.0 and one round of refinement is required. The atom-model is all heavy atom. Only the final structure is to be outputted.

Or if with this command:

$ conref – opt 3 -nopt 2 –rmsd 3.0 -cid B ex2.pdb

it means refinement is for the chain B with rmsd cutoff 3.0A and two rounds of refinement is required.

The final output is ex2_final.pdb

3) Example 3:refinement of a specified region

The third example uses conref to refine a specified region in ex3.pdb.

With this command:

$ conref – opt 3 -obj 40-60 ex3.pdb

Since the specified region contains both loop and helix regions, we use opt 3 to account for both of them.

5.8) minst

Introduction
Commands
Tutorial

Introduction

a) What is Minst?

Minst is a structure minimization program with constraints using the smoothing energy technique. The constraints include hydrogen network and backbone framework of the original structure, i.e., there is an energy penalty for the structure under minimization to deviate from the original structure or to break up a hydrogen bond that exists in the original structure. This is to make sure that the new conformation is within the similar conformational space of the original one. The constraints can be relaxed with more rounds of minimization. The smoothing energy is designed to surpass the energy barriers while enjoying the same vdw curve characteristics.

The structure minimization program can minimize protein structures in three levels: removing clashing atoms, minimization in all loop regions and minimization in all regular secondary regions plus loop regions. Energy minimization is performed using our fast torsion angle minimizer with smoothing energy techniques. Scap is applied to the structure right before it is outputted. Minimization in regular secondary structure regions is under constraints of the original hydrogen bond network. The hydrogen bond has the same definition as in the DSSP program.

Commands

For all the programs included in JACKAL package, you can find their usage by running the program with no arguments.

After unpacking jackal_*.tar.gz file that you have downloaded, you can find there is a subdirectory: jackal/minst. This directory contains all examples used in this tutorial for running minst program. Go to this directory and type:

$minst

minst will print out message about its usage as following:

```
**************************************************************************
minst is a program to minimize protein structures with constraints from the original
structure

questions? please refer to Dr.Jason Z. Xiang at zx11@columbia.edu.
**************************************************************************
Usage: minst -prm num -seed num -fast num  -opt num -nopt num -obj num-num -out
num -cid char file.pdb
-fast     fast mode.from 0-5; default is 3.with 5 fastest
-opt      minimize mode, from 1-3; default is 3.with 3 most thorough minimization
-opt   1, remove atom clashes;
-opt   2, minimize in all loop regions;
-opt   3, minimize in all loop and secondary regions
-seed     set seed number, defautl is 18120.
-prm     force field parameter mode. from 1-4, default is 3.
-prm    1, charmm all atoms;
-prm    2, amber all atoms;
-prm    3, charmm heavy atoms;
-prm   4, amber heavy atoms
-nopt     number of rounds of minimization.default is 1.
-out      output mode, from 0-2, defautl is 0. with 2 all intermediate output
-out   0, only final structures written down;
-out   1, intermediate structures also outputted;
-out   2, more intermediates output
-cid       chain id.
-obj       specify the segment to be minimized; default is whole chain.
file.pdb  pdb file
```

When you run minst, you type: minst plus option plus the integer value for the option plus the pdb file. If there is something wrong with your command, minst will exit and print out an error message. The above options can be omitted, where minst uses their default values; or several options can be used at the same time.

fast option
The option "-fast num" is used to decide how fast the minst program should run. The fast mode has 6 levels from 0 to 5 with 5 as the fastest mode. The default is 3.

seed option
Seed option is just to give a random seed number to minst program. The default is 18120.

out option
Out option is used to direct the minst program how many structures should be output, such as, if the intermediate structures should be written down to the hard disk.

When "-out 0" is used, the minst program will only write down the final model; when "-out 1" is used, the minst program will also write down intermediate structures; when "-out 2" is used, the minst program will output all intermediate structures in the process of minimization.

opt option
opt option is used to minimize the original structure. There are 3 levels of minimization with 3 as the most thorough minimization. The default level is 3. The option 1 is to remove atom clashes. The option 2 is to minimize loop regions. In the process of minimization in the helix and sheet regions, the hydrogen bonding network from the original structure is applied so that the final structure after the minimization will also observe the original regular secondary structure definition.

nopt option
The nopt option is used to decide how many rounds of minimization are required. In the first round of minimization, the minimization feels the pressure of constraints from the original structures; in the second round of minimization, the minimization is performed based on the output from the first round.

prm option
The option "-prm num" is used to decide which force field to use, either CHARMM or AMBER and which atom model to use, either the all-atom model or heavy-atom model. With option 1 or 3, the CHARMM22 force field will be used, i.e., torsion energy, van der Waals radius and charge parameters taken from CHARMM. In the case of all atom model, missing protons will first be assembled onto each residue in the protein wherever they should exist. In the case of heavy atom model, all protons on the protein will first be stripped. In the heavy atom model, charges of hydrogen atoms will give back to their parent heavy atoms so that functional group still keep the same amount of total charges.

cid option

cid option is used to specify the chain to work on. The default is the chain of empty id.

Obj option

This option is used to specify the region to work on. The default is the whole chain.

Tutorial

The tutorial of minst can be found in the directory jackal/minst

1) **Example 1: minimization of all loop regions**

The first example is the case that uses minst to minimize all loop regions.

With this command:

$ minst – opt 2 ex1.pdb

The above command line means minimization in all loop regions in the chain of empty id in ex1.pdb and one round of minimization is required. The atom-model is heavy-atom model. Only the final structure is outputted.

2) Example 2: minimization of all regions.

The second example uses minst to minimize all regions in ex2.pdb.

With this command:

$ minst – opt 3 ex2.pdb

The above command line means minimization in all loop regions in the chain of empty id in ex1.pdb and one round of minimization is required. The atom-model is heavy atom model. Only the final structure is to be outputted.

Or if with this command:
$ minst – opt 3 -nopt 2 -cid B ex2.pdb

it means minimization is for chain B and two rounds of minimization is required.

The final output is ex2_final.pdb

3) Example 3:minimization of a specified region

The third example uses minst to minimize a specified region in ex3.pdb.

With this command:

$ minst – opt 3 -obj 40-60 ex3.pdb

Since the specified region contains both loop and helix regions, we use opt 3 to account for both of them.

5.9) Util

Introduction
1. chi
2. hbond
3. nalgn
4. rotate
5. surface
6. pdbseq

Introduction

This is a collection of utility tools in the Jackal package. It includes the following software: chi, a program to print out torsion angle of a protein; hbond, a program to print out hydrogen and disulfide bond with DSSP definition; nalgn, a program to superimpose two protein structures based on a given sequence alignment; pdbseq, a program to print out the sequence information of a protein; rotate, a program to manupilate a protein structure with torsion angles given in a file; surface, a program to calculate solvent accessible surface.

The following will briefly explain these programs and their tutorials:

1. chi

Chi is a program to print out torsion angle of a protein. It has the following interface:

Usage:
chi -a num –o str file.pdb
-a num: options for torsion angle output.default is 0
-a 0: print out all torsion angles.
-a 1: print out only backbone phi,psi and sidechain torsion angles.
-a 2: print out only backbone phi,psi torsion angles.
-a 3: print out only sidechain torsion angles.
-a 4: print out only backbone omega torsion angles.
-o output file. Optional. Default is standard screen output.

With option "-a 0", chi program will print out all torsion angles including omega.

For example, the following command on ex1.pdb:

$chi –a 0 –o ex1.chi ex1.pdb

The output s is as follows:

```
!             OMEGA   PHI     PSI      X1       X2      X3
T    1         999.0   999.0   144.8   -60.2
T    2         177.2  -110.5   147.0    57.2
C    3        -178.9  -133.4   137.9   -51.7
C    4        -177.0  -122.3   147.8   -67.4
P    5        -175.5   -77.2   -18.6
S    6        -179.3  -156.3   167.6    70.1
I    7         178.2   -60.6   -48.6   -72.3   179.7
V    8         179.0   -58.4   -40.3   153.8
A    9         179.2   -61.0   -41.8
```

The first column is the residue name and the second name is sequence id. The following columns are for omega, phi, psi, x1, x2, x3, x4 and x5 respectively. The value “999” means undefined.

With the following command:

$ chi –a 2 ex1.pdb

This command will print out only phi and psi torsion angles:

```
!              PHI     PSI
T    1        999.0   144.8
T    2       -110.5   147.0
C    3       -133.4   137.9
C    4       -122.3   147.8
P    5        -77.2   -18.6
S    6       -156.3   167.6
I    7        -60.6   -48.6
V    8        -58.4   -40.3
A    9        -61.0   -41.8
R   10        -63.3   -45.2
S   11        -59.3   -41.6
N   12        -66.8   -37.6
```

2. hbond

Hbond program can be used to print out hydrogen bond and disulfide bond information of a protein. Hydrogen bond is defined by DSSP.

Its interface:

Usage:
hbond -h num -o str file.pdb
-h num: options for output hydrogen bond.default is 0
-h 0: hydrogen bond plus s-s bond.
-h 1: hydrogen bond only.
-h 2: s-s bond only.
-o output file.optional. default is standard screen output

For example, with the command:

$hbond –h 0 ex1.pdb

this will print out all hydrogen and disulfide bonds:

```
!RES ID ATM RES ID  ATM    DIS
T    1  O   I   35  N      2.83
T    1  OG1 G   37  O      2.72
```

```
T     2   O    R    10   NH2     2.94
T     2   OG1  R    10   NH1     2.87
C     3   N    I    33   O       2.91
C     3   O    I    33   N       2.84
C     4   N    N    46   O       2.92
C     4   O    N    46   N       2.95
S     6   O    R    10   N       3.14
I     7   O    S    11   OG      3.30
I     7   O    S    11   N       2.88
V     8   O    N    12   N       2.84
A     9   O    F    13   N       2.86
R    10   O    N    14   N       2.90
R    10   NH2  N    46   O       2.98
S    11   O    V    15   N       3.13
N    12   O    C    16   N       2.91
F    13   O    R    17   N       2.94
N    14   O    L    18   N       3.40
V    15   O    L    18   N       2.99
C    16   O    T    21   N       4.00
R    17   O    G    20   N       2.80
R    17   NH1  T    21   O       2.81
P    22   O    C    26   N       3.03
E    23   O    A    27   N       2.76
A    24   O    T    28   N       3.22
L    25   O    T    28   OG1     3.26
L    25   O    Y    29   N       2.97
C    26   O    T    30   N       3.05
A    27   O    T    30   OG1     3.50
A    27   O    G    31   N       2.85
T    30   OG1  C    32   N       3.09
A    38   O    C    40   N       3.84
P    41   O    Y    44   N       2.95
C     3   SG   C    40   SG      2.03
C     4   SG   C    32   SG      2.04
C    16   SG   C    26   SG      2.03
```

In the above printout, the last column is the distance between the corresponding atoms. The last three lines are about disulfide bonds.

3. nalgn

This program is to superimpose two structures based on a given sequence alignment. The alignment is in pir format of nest program.

Its interface:

Usage:
nalgn -f num -o str file.pir
-f num: options for algnment two structures.default is 0
-f 0: ca only.
-f 1: backbone only.

-f 2: all heavy atoms.
-f 3: all atoms including protons.
-o output file.

For example:

$nalgn –f 0 –o a.pdb ex1.ali

The above command says aligning two structures with the given sequence alignment specified in file ex1.ali. The two structures are specified in the structure pirs. The following is an example:

```
>P1;1hbaA
structureN:1hba:0:A:141:A:::-1.00:-1.00
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFD--L----
SH-GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLL
SHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLT-SK-YR*
>P1;SEQ
structure:myg: 1 : A :150: A :::-1.00:-1.00
G-SDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKF-DKFKHLKSED
EMKASEDLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKIPVKYLEFI
SEAIIQVLQSKHPGDFGADAQGAMSKALELFRNDMAAKYKLG*
```

ex1.ali specifies two pir alignments, both of them are of structure token. This means superimposing chain A in 1hba.pdb with chain A in myg.pdb according to the given sequence alignment between the two chains. In superimposition, only alignments corresponding to no-gap residues are superimposed; otherwise, they are ignored. For example, the second residue in chain A of 1hba is Leu, which corresponds to a gap in the other pir alignment of myg. So L2 in chain A of 1hba will not be considered in the structure superimposition of the two chains.

However, if with this command:

$ nalgn –f 3 –o a.pdb ex1.ali

The command means structure superimposition with all atoms. Some atoms may exist in one residue of a chain but do not exist in its corresponding residue in the other chain. Only those atoms that exist in both residues are considered.

4. pdbseq

It has the following interface:

Usage:
pdbseq -a num -c char -o str file.pdb
-a num: options for reading sequence card. default is 0
-a 0: read sequence from sequence card in pdb.
-a 1: read sequence from ATOM record including gap with X.

-a 2: read sequence from ATOM record including no gap.
-c char: one character for the chain to be read.default is all chains.
-o output file.optional. default is standard screen output

5. rotate
The program is used to rotate bond according to given angles.

Usage:
rotate -d num -e num -t name -n num -r num file.pdb
-t name: chiangle file name. required.
-d num: the direction of rotation. default is 1
-d 1: rotation forward.
-d 0: rotation backward.
-e num: the end residue where rotation affects. default is to the end
-n num: the content of chiangle file.
-n 0: the chiangle file includes omega, phi,psi, side-chain torsion angle
-n 1: the chiangle file includes phi,psi, side-chain torsion angle
-n 2: the chiangle file includes phi,psi
-n 3: the chiangle file includes side-chain torsion angle
-r num: the option to rotate the bond.default 1
-r 0: rotate the bond of degrees specified in chi file.
-r 1: rotate the bond so that its torsion angle is equal to that specified in chi file.

Option "–t" is used to specify a file name which should contain torsion angle. The torsion angle is used for rotate program to decide how much the bond should rotate; option "-n" is used to tell the content of the file specified by option "-chi"; option "-r" is used to specify how rotate program should use the chi angle in the file specified by "-t" option; option "-d" is used to tell rotate program the direction of rotation, either forward or backward; option "-e" is used to decide the end residue rotate program will affect.

For example:

$rotate –chi ex1.chi –n 0 –r 1 ex1.pdb

The above says that the chi angle file is ex1.chi, which contains information about angles in sequential order of omega, phi, psi, x1, x2 and so on…Option "-r 1" means rotating the corresponding bond until its torsion angle is equal to that specified in ex1.chi file. The output is ex1_rotate.pdb.

The ex1.chi is as follows:

```
T    2       177.2  -110.5   147.0    57.2
C    3       999.0  -133.4   137.9   -51.7
C    4      -177.0    22.3   147.8   -67.4
P    5      -175.5   -77.2   999.0
S    6      -179.3  -156.3   167.6    70.1
I    7       178.2   -60.6   -48.6   -72.3   179.7
V    8       179.0   -58.4   -40.3   153.8
```

```
A    9        179.2   -61.0   -41.8
R   10        176.5   -63.3   -45.2   177.7
S   11        179.3   -59.3   -41.6   -61.0
```

In file ex1.chi, rotate program will ignore any values specified above 360. So omega value of 999 for C3 and psi value of 999 for P5 are ignored, i.e., the omega for C3 and psi for P5 will keep its original angle after rotation. After rotation, the protein will be in a new conformation so that every torsion angle will be equal to that specified in the ex1.chi file, e.g., omega, phi, psi and x1 of T2 will be equal to 177.2, -110.5, 147.0 and 57.3 respectively. In the above example, R9 does not have all its torsion angle specified, it has only four values, which are four omega, phi, psi and x1 respectively. So after rotation, x2, x3, x4 and x5 will not change.

However, if using the following command:

$rotate –chi ex1.chi –n 0 –r 0 ex1.pdb

which means rotate clockwise by the angle specified in ex1.chi. For example, phi of C4 will rotate clockwise 22.3 degree. If the original value is 100, the new phi angle after rotation will be 122.3.

If with the following command:

$rotate –chi ex1.chi –n 2 –r 0 ex1.pdb

The option "-n 2" means the chi angle file ex1.chi contains only phi and psi values, so any values sets beyond that are ignored. The above example will use 177.2 and –110.5 for phi and psi value respectively, and 177.2 will not be considered for omega anymore and other values after –110.5 are also ignored.

If in the case of multiple chains, the chi angle file should be in the format, for example:

```
V    1 A     999.0   999.0   171.0  -157.8
L    2 A     178.7   -76.8   126.5   -72.1  -169.7
S    3 A     179.7   -87.7   165.8    74.7
P    4 A     179.3   -52.8   -42.1
F  117 C    -176.7   -85.3    51.0  -161.7    71.5
T  118 C     178.9   -70.1   163.9    67.3
P  119 C     178.1   -52.9   -42.7
A  120 C    -178.5   -69.9   -33.2
V  113 D     178.8   -66.3   -44.9   160.2
L  114 D     178.2   -61.5   -36.9   -67.2   174.0
A  115 D    -179.4   -69.0   -41.6
H  116 D     179.8   -52.5   -50.0  -171.7    61.7
H  117 D     179.8   -72.6   -47.5   -88.0   -89.4
```

In multiple chains, the third column is for chain id.

6. **surface**

The program is used to print out solvent accessible surface. It has the following interface:

Usage:
surface -prm num -probe float -size num -out str file.pdb
-prm: set parameter set.
-prm 1: Charmm22 all atom model
-prm 2: Amber94 all atom model
-prm 3: Charmm22 heavy atom model
-prm 4: Amber94 heavy atom model
-probe: set probe radius. default is 1.4 A
-size: number of subareas each atom surface subdivided.
-out output file.default is standard screen output